# COOPERATIVE LOCALIZATION ON COMPUTATIONALLY CONSTRAINED DEVICES

THESIS

Randy S. Cicale II, Captain, USAF

AFIT/GCO/ENG/12-04

**DEPARTMENT OF THE AIR FORCE**
**AIR UNIVERSITY**

***AIR FORCE INSTITUTE OF TECHNOLOGY***

**Wright-Patterson Air Force Base, Ohio**

AFIT/GCO/ENG/12-04

# COOPERATIVE LOCALIZATION ON COMPUTATIONALLY CONSTRAINED DEVICES

THESIS

Presented to the Faculty

Department of Electrical and Computer Engineering

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

In Partial Fulfillment of the Requirements for the

Degree of Master of Science

Randy S. Cicale II, BS

Captain, USAF

March 2012

AFIT/GCO/ENG/12-04

# COOPERATIVE LOCALIZATION ON COMPUTATIONALLY CONSTRAINED DEVICES

Randy S. Cicale II, BS
Captain, USAF

Approved:

[illegible]
Jeffrey Hemmes, Maj, USAF (Chairman) — 28 Feb 12 Date

[illegible]
Todd R. Andel, Maj, USAF (Member) — 2 MAR 12 Date

[illegible]
Thomas Dube, Maj, USAF (Member) — 2 MAR 12 Date

AFIT/GCO/ENG/12-04

## Abstract

Cooperative localization is a useful way for nodes within a network to share location information in order to better arrive at a position estimate. This method is useful in GPS contested environments such as indoors and urban settings. Most research or commercial systems exploring cooperative localization rely on either special hardware, or extra devices to store the database or make computations. Research also deals with specific localization techniques such as using Wi-Fi, ultra-wideband signals, or accelerometers independently opposed to fusing multiple sources together.

This research brings cooperative localization to the smart phone platform, specifically to take advantage of the multiple sensors commonly available. The entire system runs on Android powered devices, including the wireless hotspot. In order to determine the merit of each sensor, analysis is completed to determine successes and failures. For this research, the accelerometer, compass, and received signal strength capability are examined to determine their usefulness in cooperative localization.

The result of the research effort is software written for the Android platform that can improve on a location estimate. By conducting an experiment at meter intervals for 7m, the system detected changes in location at each interval with an average standard deviation of 0.44m. The closest location estimates occur at 3m, 4m and 6m distances with average errors of 0.15m, 0.11m, and 0.07m respectively. This outcome indicates that precise estimates are achieved with an Android hotspot and mobile nodes.

## Acknowledgments

I would like to thank my thesis adviser, Maj Jeffrey Hemmes, for his support during this work. His guidance and optimistic attitude has kept me focused and working hard to realize my goal. I would also like to thank Maj Todd Andel for his support and assistance throughout the process as well. A special recognition is deserved for Maj Thomas Dube who was able to fill in last minute as my committee member. The brief time we were able to work together makes me wish we could have collaborated much earlier on in this process.

My loving wife and children deserve a great deal of gratitude as they continually provided support and encouragement throughout the entire process. Their sacrifices through late nights and stressful times did not go unnoticed and are greatly appreciated. This effort would not be what it is without their love.

Randy S. Cicale II

## Table of Contents

Page

Abstract ... iv

Acknowledgments ... v

Table of Contents ... vi

List of Figures ... viii

List of Tables ... x

List of Equations ... xi

I. Introduction ... 1

1.1 Motivation ... 1
1.2 Computationally Constrained Devices ... 2
1.3 Cooperative Localization ... 3
1.4 Research Goals ... 4
1.5 Methodology ... 5
1.6 Assumptions and Limitations ... 5
1.7 Implications ... 6
1.8 Thesis Overview ... 6

II. Literature Review ... 8

2.1 Chapter Overview ... 8
2.2 Android Powered HTC Hero ... 8
2.3 Cooperative Localization ... 14
2.4 RSS and Fingerprinting ... 18
*2.4.1 Lateration Based Algorithms ... 22*
*2.4.2 Classification Based Algorithms ... 22*
2.5 Data Fusion ... 23
2.6 Improving Accuracy ... 27
*2.6.1 WLAN and Bluetooth ... 27*
*2.6.2 WLAN and Accelerometer ... 28*
*2.6.3 WLAN and Signals of Opportunity ... 29*
2.7 Summary ... 30

III. Methodology ... 31

3.1 Chapter Overview ... 31
3.2 Problem Definition ... 31

*3.2.1 Goals and Hypothesis* ........ *31*
*3.2.2 Approach* ........ *32*
3.3 System Setup ........ 33
3.4 System Services ........ 40
3.5 Workload ........ 41
3.6 Performance Metrics ........ 42
3.7 System Parameters ........ 43
*3.7.1 Workload Parameters* ........ *43*
*3.7.2 System Parameters* ........ *44*
3.8 Evaluation Technique ........ 45
3.9 Experimental Design ........ 46
3.10 Summary ........ 47

IV. Analysis and Results ........ 48

4.1 Chapter Overview ........ 48
4.2 Results of Experiment Scenarios ........ 48
*4.2.1 RSS Observations* ........ *49*
*4.2.2 Accelerometer* ........ *59*
*4.2.3 Compass* ........ *62*
*4.2.4 Overall Localization System* ........ *63*
*4.2.5 Worst Case Scenarios for Distance Estimation* ........ *70*
4.3 Investigative Questions Answered ........ 72
4.4 Summary ........ 73

V. Conclusions and Recommendations ........ 74

5.1 Chapter Overview ........ 74
5.2 Conclusions of This Research ........ 74
5.3 Significance of Research ........ 75
5.4 Recommendations for Future Research ........ 76
5.5 Summary ........ 77

Appendix A ........ 79

Appendix B ........ 81

Bibliography ........ 82

## List of Figures

Page

Figure 1 - The major components of the Android Operating System (Google, 2010). .... 10

Figure 2 - Sample timeline for finding user location (Google, 2010) .............................. 11

Figure 3 - Triangulation typically used in cooperative localization when multiple access points are available ........................................................................................ 21

Figure 4 - JDL Data Fusion model, slightly revised for cooperative localization ............ 26

Figure 5 - SELFLOC system (Gwon, Jain, & Kawahara, 2004) ...................................... 28

Figure 6 - Screenshot of localization application ............................................................ 37

Figure 7 - Example communication between nodes ........................................................ 42

Figure 8 - Simple testing diagram showing how experiment will be setup ...................... 46

Figure 9 - Observed RSS values in hallway ..................................................................... 49

Figure 10 - RSS recordings in hallway; 2 min at each location....................................... 50

Figure 11 - 2 min averages in hallway with unpredictable results after 4m ..................... 51

Figure 12 - RSS recordings in hallway with another phone showing inaccurate readings52

Figure 13 - 2 min averages in hallway showing averaging does not reduce the inconsistencies of the results ....................................................................... 52

Figure 14 - RSS recordings in a lab environment with a general logarithmic decay ....... 53

Figure 15 – 2 min averages of the previous plot where the trend line does not suggest a realistic path loss exponent........................................................................... 54

Figure 16 - RSS recordings in a laboratory using a different phone ................................ 55

Figure 17 - 2 min average of the previous plot with a much more realistic logarithmic decay ........................................................................................................... 55

Figure 18 - Two most similar runs compared showing a similar drop in RSS values at different distances ........ 56
Figure 19 - RSS recordings used for building the database ........ 57
Figure 20 - Averaged RSS values with standard deviations ........ 58
Figure 21 - Expected path-loss curve estimates for 3 exponents compared to actual recordings ........ 59
Figure 22 - Acceleration X and Y values moving right to left to right ........ 60
Figure 23 - Acceleration X and Y values moving phone up and down ........ 61
Figure 24 - Acceleration X and Y values moving phone in quarter circle ........ 62
Figure 25 - Compass values with interference added ........ 63
Figure 26 - Estimations at 0.254m show the system over-estimates the position ........ 64
Figure 27 - Estimations at 1m were split between near 1m or 2m ........ 65
Figure 28 - Estimations at 2m overestimate the distance ........ 65
Figure 29 - Estimations at 3m are fairly close to the actual distance ........ 66
Figure 30 - Estimations at 4m were close to the actual distance ........ 66
Figure 31 - Values at 5m are underestimated ........ 67
Figure 32 - Estimations at 6m were closest to the actual distance ........ 68
Figure 33 - Estimations at 7m were all underestimates ........ 68
Figure 34 - Error plot for each distance value showing if the average estimate was an over or under estimate and how widespread the standard deviation was ........ 69
Figure 35 - Worst Case Estimations for 0.254m, 2m, 4m, and 6m ........ 71

## List of Tables

Page
Table 1 - HTC Hero Capabilities and Limitations ........ 13

Table 2 - Motorola Droid Capabilities and Limitations ........ 14

Table 3 - Summary of Signaling Techniques ........ 16

Table 4 - Path Loss Exponents for Different Environments (Rappaport, 2002) ........ 19

Table 5 - Summation of Techniques and Error Ranges ........ 21

Table 6 - Example GPS transmission ........ 38

## List of Equations

Page

Equation 1 .......... 18

Equation 2 .......... 20

Equation 3 .......... 20

Equation 4 .......... 22

Equation 5 .......... 42

# COOPERATIVE LOCALIZATION ON COMPUTATIONALLY CONSTRAINED DEVICES

# I. Introduction

## 1.1 Motivation

An elite U.S. Army Special Forces group is tasked to extract a high value target from a building. As the group navigates through the dark building, shots are fired. In order to maintain silence, the soldiers check their standard issue smart phones to see the locations of other members. Knowing one of their fellow soldiers is beyond the wall where the shots were heard, they carefully enter the room and take aim away from the indicator on their phone and apprehend the target. This fictitious scenario may not be far from the future as a December 2010 article in the Army Times (Gould & Hoffman, 2010) reports that the Army is seeking to modernize its force by issuing every soldier a smart phone. The phones are envisioned to provide soldiers with real-time intelligence and video from unmanned vehicles. With combat operations in urban environments, some of the traditional tracking techniques such as Global Positioning System (GPS) and cellular tower triangulation are unavailable. In order to track positions of soldiers in buildings or without the previously mentioned services, another method needs to be utilized. The ability for a soldier to determine if a friendly force is in the next room could mean the difference between life and death in a combat situation.

The previous scenario may describe just a tactical military situation, but as Nielsen ratings report, 40% of all mobile users own a smart phone (Kellogg, 2011). As the special edition IEEE Signal Processing Magazine (Sun, Chen, Guo, & Liu, 2005) describes, a mandate exists to refine E-911 location estimates using cellular tower triangulation. But what would happen when a natural disaster strikes and the tower is unable to triangulate a user's position, or if the host nation infrastructure is unreliable? As they may be stuck under rubble, a program running on the individual's phone may help rescue workers efficiently locate their whereabouts and save the person's life.

## 1.2 Computationally Constrained Devices

Moore's Law states that the complexity for minimum component costs increases at a rate of roughly double every two years (Moore, 1965). In other words, devices get faster and cheaper allowing for them to also shrink in size. Recent years have seen an explosion of this technological feat as handheld devices have become cheap enough for the average consumer to afford. The term computationally constrained no longer carries the same connotation. Devices at the fraction of the size of their fully functioning counterpart (i.e., phones vs. computers), still possess all of the functionality of their larger brethren, albeit at the speed of perhaps a few years ago. Smart phones, which have a growing market share of the mobile industry, currently have 1GHz processors with 512MB of RAM and 16GB of hard drive space. Cutting edge versions sport a dual core processor, 1GB of RAM and over 32GB of hard drive space. The number of these phones in the U.S. exceeds 72.5 million of the 234 million Americans that use cellular phones (Flosi, 2011). This emergence allows for advanced tasks to be executed while maintaining mobility. These phones are no longer for making online phone calls; they can edit a variety of document types, play games, access the web, and more all at the same time.

The operating systems that run these devices are not lacking in capability either. The two dominant systems on the market today are Google’s Android and Apple’s iOS. Android is based on the Linux kernel and is open source allowing for full customization. Apple’s iOS, however, is proprietary and derived from the Mac OS X operating system. Because of the open nature of Android, it was chosen as the operating system for this research. If a system service needs to be enhanced, or tweaked, the Android kernel can be modified and recompiled. This operating system is also making its way onto other devices such as watches, televisions and tablets allowing for a greater expansion of targeted users.

## 1.3 Cooperative Localization

Cooperative localization is the ability for a node to determine its location relative to other nodes. The term node can refer to a device, such as a laptop or smart phone, which has wireless capabilities. Nodes communicate location estimates to one another in order for other nodes to know where it is. This assistance among nodes helps each obtain a much better accuracy than if no nodes worked together. Popular uses for cooperative localization are in the field of robotics for autonomous navigation and in logistical systems for locating assets. Nodes typically determine their location with the help of anchor nodes, which know their exact location. For example, if a node is receiving signals from three anchor nodes, it can triangulate its position and relay it to other nodes that do not know where they are. In practical localization scenarios, cooperative localization is often not the primary means of localizing nodes, but rather is used to augment services such as GPS and others when they are not available.

## 1.4 Research Goals

This thesis lays the groundwork of bringing cooperative localization to sensor-rich, computationally-constrained devices, such as smart phones. By using multiple sensors, a cooperative localization algorithm can produce more robust results because errors in any one sensor reading can be corrected by another.

The goals of this research are to:

- Bring real-time cooperative localization to computationally constrained devices by using their Wi-Fi hotspot capability, Bluetooth, accelerometer and other sensors available to the device.
- Demonstrate this ability on commercial-off-the-shelf (COTS) hardware and examine strengths and weaknesses of the hardware.
- Organize the data fusion with guidance from fusion methods not previously applied to cooperative localization.

These goals are realized by producing a proof-of-concept application can indicates neighbor nodes with improved accuracy over methods that do not utilize the various sensors. While the results of this thesis offer an improvement over previous cooperative localization schemes, the final product is not a fully optimized version which takes into consideration battery life and security. Communication between the devices will be in the most direct and efficient manner possible and not take into account malicious users.

## 1.5 Methodology

The realization of this research is an iterative process. After sufficient background material is gathered on the topic of cooperative localization, the ability for a smart phone to support the process is assessed. In support of that effort, the software development kit (SDK) need to be explored to understand how the Android application programmer interface provide core services for application development. Once the abilities of the phone are understood, a method will be developed to combine the various sensors to accurately estimate the phone’s location within a small network of other phones. For the location estimation to work effectively, a database needs to be implemented to store distances that correspond to the received signal strength. As the application develops, testing is done to determine the effectiveness of a given measurement routine.

## 1.6 Assumptions and Limitations

In order to execute this thesis, several key assumptions are made. As nodes stay on, their batteries deplete which can cause fluctuations in transmit power and degraded processing capability as the host operating system attempts to conserve resources. These fluctuations are controlled by testing the system with full battery capacity and limiting the duration of the tests so the battery does not fall below 50% of full capacity. Another assumption is that any GPS data for a node that might adjust its position is considered stale. In other words, no current, active GPS readings are used for the purpose of localizing nodes. The primary goal of this research is to localize nodes without the use of active GPS data. By isolating the capabilities and contributions of this research, it could then be applied to systems in an effort to augment localization when GPS or cell tower triangulation is unavailable.

Other key assumptions rely on the lower layers of the Open Systems Interconnection (OSI) model. The physical layer is assumed to not be ideal. Since the system may be tested indoors or outdoors, there is no way to create a perfect, interference-free physical area. Therefore, the device must accurately handle collision avoidance and follow proper IEEE 802.11 signaling techniques to minimize packet loss. Another physical layer assumption is bidirectional communication; that is if node A can transmit to node B, then node B can also transmit to node A. Also, as stated before, the communication will not be encrypted. All nodes are assumed to be trustworthy to simplify the effort and focus on the goal of localizing nodes within the network.

## 1.7 Implications

When this thesis is realized, the ability to estimate locations of nearby users will be more accurate than if trying to determine the location non-cooperatively. If every soldier is equipped with a smart phone, they can use the application to keep track of each other whether the battlefield is in the open, in GPS-limited environments, or completely cut-off from GPS such as indoors. This capability or technology could reduce friendly fire and aid in search and rescue. The application does not require cellular service and can be adjusted based on what approximation techniques the user chooses.

## 1.8 Thesis Overview

The remainder of this thesis is organized as follows. Chapter 2 explores the background of Android and the capabilities of the operating system. It also discusses the previous work done in the field of cooperative localization to include common implementation approaches.  Following the background, chapter 3 details the methodology used to design, setup, and conduct the experiment to determine the feasibility of using an Android smart phone to perform cooperative localization. Chapter 4 discusses the results along with an analysis to quantify how well the system performs. Finally, chapter 5 concludes the work and provides suggestions for future work.

# II. Literature Review

## 2.1 Chapter Overview

This chapter presents an overview of prior research. First, the capabilities of Android devices have to be determined, followed by cooperative localization basics. From there, the finer implementations of cooperative localization are discussed such as received signal strength (RSS), measurements and location fingerprinting. Section 2.2 provides information about the HTC Hero provided for the experiment as well as the Android operating system. Section 2.3 explores cooperative localization techniques while Section 2.4 explains that using RSS is the most feasible technique because it does not require special hardware or timing abilities. Following that, Sections 2.5 and 2.6 explore techniques to increase location estimation accuracy. Section 2.7 summarizes the chapter and leads the way forward.

## 2.2 Android Powered HTC Hero

In coordination with the Open Handset Alliance (OHA), Google released the Android mobile operating system in 2007 with the first commercially available phone released in October 2008. The OHA consists of over 90 companies committed to the development of open standards for mobile devices (Alliance, 2007). According to Nielsen ratings, Android had less than 5% of the mobile operating system market share in 4Q09. Just one year later it has reached 19% (Google, 2010). The success is in large part due to the diverse hardware and the Android Market which has over an estimated 500,000 applications available with developers ranging from multinational software corporations to hobbyists coding in their spare time.

Android is based on the Linux Kernel ranging from 2.6.27 (Android version 1.5, API level 3) to 2.6.35 (Android version 2.3.4, API level 10) and most recently 3.0.1 (Android version 4.0.3, API level 16). It is useful to note that there is also a version 3.1 of the OS; however, at the time of this research that platform is reserved for tablet devices. In late 2011, the version 2.x series and the 3.x series merged into 4.x to combine tablet and cell phone programming functionality.

The Android kernel handles device drivers, memory management, process management and networking. Figure 1, from the Android Developer website details how the framework is organized and what libraries are available. The diagram shows the kernel at the bottom layer which consist of the Android native libraries written in C and C++. Then those libraries are incorporated with Java native interfaces. After the interfaces is the Dalvik Virtual Machine responsible for running the Java-implemented application layer (Google, 2010) (Shabtai, Fledel, Kanonov, Elovici, Dolev, & Glezer, 2010) (Enck, Ongtang, & McDaniel, 2009). For the most part, developers utilize the Application Framework to invoke the libraries underneath, i.e., a program that tracks your path while running would invoke the `LocationManager`.

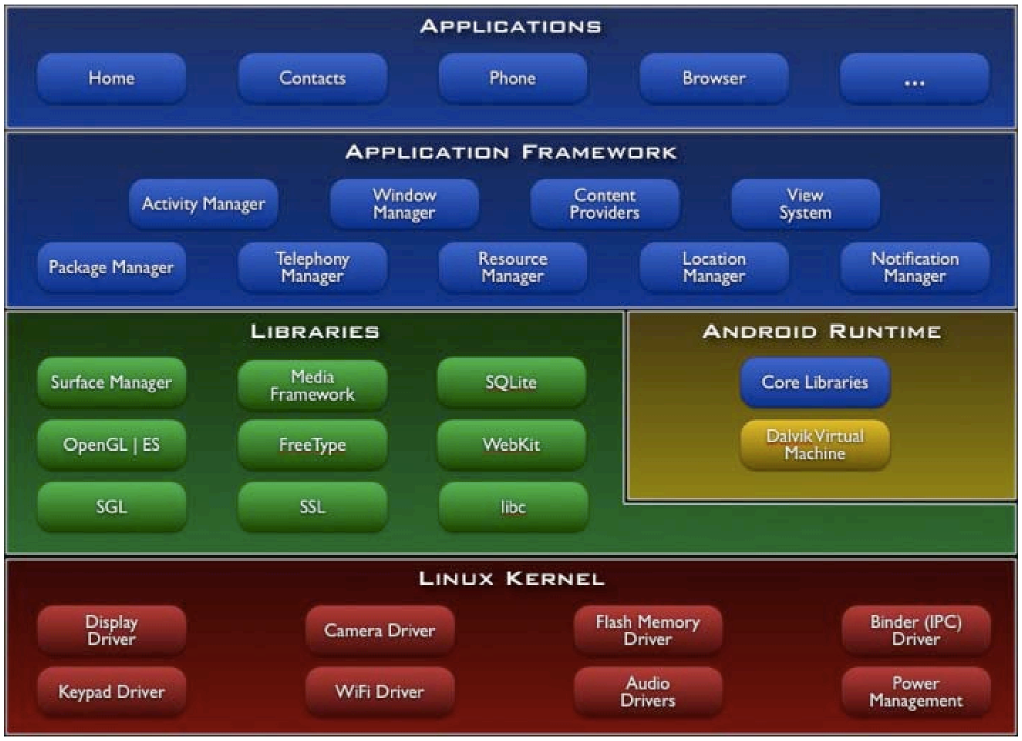


**Figure 1 - The major components of the Android Operating System (Google, 2010).**

In an effort to make applications more useful to the user, Google provides an in-depth developers guide on their website http://developer.android.com. Applications are divided into four main components: activities, services, content providers and broadcast receivers. Activities are graphical screens that are displayed to the user; different screens that the user could select would represent different activities in the program. Services are components that run in the background and can perform long-running tasks. Services can be bound to the activity and therefore end when it ends, or can run after the activity has been closed. Content providers allow the programmer to share application data with other programs. Finally, broadcast receivers respond to system-wide announcements. These can be used to pass information from one service to another, or alert the program when the screen goes blank. Another feature that can be used is creating a single "application" file. This can tie the whole application together and house commonly used functions.

Since location awareness is becoming more popular in mobile devices, a guide has been written to specifically address this issue. While the suggested program flow for obtaining the users location is coarse for this research, it serves as a starting point. The Android development guide suggests to listen for updates from the desired location provider (in this instance, Wi-Fi); maintain a "current best estimate" of location by filtering out new, but less accurate fixes; stop listening for location updates; and finally take advantage of the last "best" location estimate (Google, 2010). Figure 2 depicts a sample timeline. Balancing when to start and stop listening for location updates requires making sacrifices between battery life and location accuracy. The Java class responsible for managing the location capabilities is aptly named `LocationManager`.

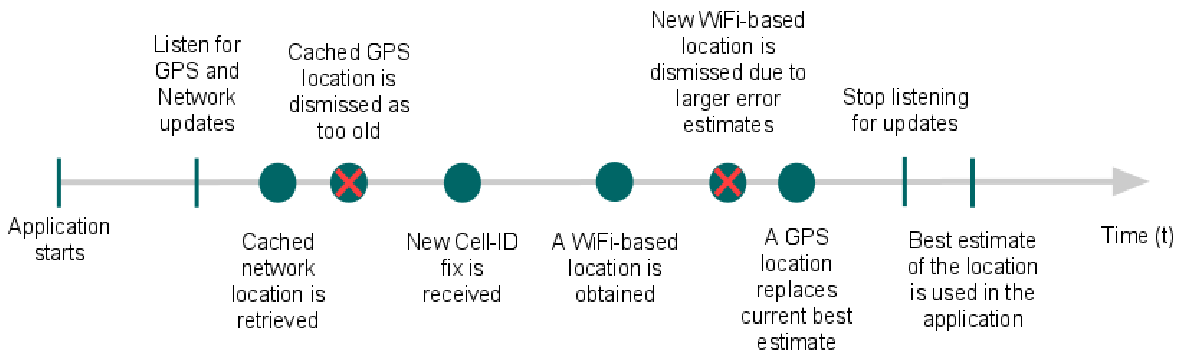

Listen for GPS and Network updates
Cached GPS location is dismissed as too old
New WiFi-based location is dismissed due to larger error estimates
Stop listening for updates
Application starts
Cached network location is retrieved
New Cell-ID fix is received
A WiFi-based location is obtained
A GPS location replaces current best estimate
Best estimate of the location is used in the application
Time (t)

**Figure 2 - Sample timeline for finding user location (Google, 2010)**

The "`android.location`" package calls the `LocationManager` system service, which provides APIs to determine location and bearing for the device. Calling this instance allows the program to query for the list of all `LocationProviders`, a class that tracks the last known user location, registers/unregisters for periodic updates of the user's current location from a location provider, and registers/unregisters for a given `Intent` to be fired if the device comes within a given proximity of a given latitude/longitude. This information can be combined with the Maps package included with the OS. This functionality is limited to GPS location tracking and cell tower triangulation capabilities, but could be beneficial for this research.

Other valuable location sensors that can be used in determining location with respect to another device are a compass and accelerometer. The compass is accessed from the `GeomagneticField` class which produces the declination, inclination, field strength and X, Y, Z values. This information is useful for determining the traveling direction to or from a friendly node. The accelerometer is accessed by using the `SensorEvent` API and declaring a Sensor type object of `Accelerometer`. This sensor event will hold the time-stamp, accuracy and the data of the sensor. As described in the `SensorEvent` API, the phone's X-axis is on the horizontal plane corresponding to the right and left of the phone. Acceleration to the right should register positive values and to the left should register negative values. Similarly, the Y-axis is vertical plane corresponding to the top and bottom of the phone. The Z-axis points toward the outside of the screen and a "falling" movement should register negative values.

For this research, HTC Hero and Motorola Droid phones are used, which are rooted and upgraded to Android 2.3.7 for the Wi-Fi hotspot capability. The HTC phone is equipped with the Qualcomm MSM7200A chipset which includes support for 802.11 b/g, digital compass and Bluetooth v2.0 (HTC, 2011). The Motorola phone is equipped with the TI OMAP 3430 chipset which includes support for 802.11 b/g, digital compass and Bluetooth v2.1 (Motorola, 2009). Tables 1 and 2 list the capabilities of each phone. While Bluetooth has been used successfully in cooperative localization (Gwon, Jain, & Kawahara, 2004), the Android Bluetooth API does not provide a mechanism for measuring received signal strength. The ability to use Bluetooth as a location device could possibly be achieved by modifying the kernel.

**Table 1 - HTC Hero Capabilities and Limitations**

| Sensor | Chipset | Specifications |
|---|---|---|
| **Wi-Fi** | Qualcomm MSM7200A | +802.11 b/g |
| **Bluetooth** | Qualcomm MSM7200A | -Version 2.0 + EDR |
| **Accelerometer** | Bosh BMA 150 | +25-1500Hz Bandwidth<br>+3000Hz Refresh Rate<br>-500 μg/√Hz Acceleration Noise Density |
| **Magnetometer** | Asahi Kasei AK8973 | +12.6ms Time for Measurement<br>+/-2.0mT Offset Magnetic Field Compensation |
| **GPS** | Qualcomm MSM7200A | +Enhanced filtering software to optimize accuracy<br>+gpsOneXTRA for enhanced standalone performance |
| **Other Relevant Information** | Qualcomm MSM7200A | -CPU is 528 MHz ARM11<br>-288 MB RAM |

**Table 2 - Motorola Droid Capabilities and Limitations**

| Sensor | Chipset | Specifications |
|---|---|---|
| **Wi-Fi** | TI OMAP 3430 | +802.11 b/g |
| **Bluetooth** | TI OMAP 3430 | -Version 2.1 + EDR |
| **Accelerometer** | LIS331DLH | +500-1000 Hz Bandwidth<br>-218 µg/√Hz Acceleration Noise Density |
| **Magnetometer** | Asahi Kasei AK8973 | +12.6ms Time for Measurement<br>+/-2.0mT Offset Magnetic Field Compensation |
| **GPS** | TI OMAP 3430 | +aGPS and sGPS |
| **Other Relevant Information** | TI OMAP 3430 | -CPU is 600 MHz ARM Cortex A8*<br>-256 MB RAM |

*Chipset advertises 3x performance gain over the ARM11 used in the HTC Hero

## 2.3 Cooperative Localization

The Encyclopedia of Geographical Information Sciences (GIS) defines cooperative localization as

> "The estimation of the locations of wireless devices (aka nodes) in a network using measurements made between many pairs (or subsets) of nodes. While many localization methods limit an unknown-location device to making measurements with known-location nodes, cooperative localization methods specifically encourage measurements to be made between nodes regardless of each node's prior location knowledge. Then, cooperative localization algorithms use the 'mesh' of measurements to simultaneously estimate the coordinates of all nodes." (Patwari, Localization, Cooperative, 2008)

The signal used to determine the location is calculated by Time of Arrival (TOA), Angle of Arrival (AOA), Time Difference of Arrival (TDOA), or Received Signal Strength (RSS). Since the AOA method of localization requires steering the main lobe of an adaptive phased array antenna in the direction of an arriving signal (Sun, Chen, Guo, &

Liu, 2005), it is not a viable solution for smart phone cooperative localization. The TOA and TDOA methods estimate the time required for a signal to go from the transmitter to the receiver. This typically relies on line-of-sight (LOS) measurements and is prone to multipath, which is the diffraction of signal through the environment. Because the operational environment for this research may range from indoor to outdoor and urban environments, using TOA or TDOA measurements may not produce the best results. Work done by (Gustafsson & Gunnarsson, 2005) and (Patwari, Ash, Kyperountas, III, Moses, & Correal, 2005) compare AOA, TOA, TDOA, RSS and RSS map-based positioning and determined that RSS is least informative followed by AOA. When an RSS map is used, it performs at the same level as AOA and approaches TOA/TDOA. RSS also has the capability of outperforming TOA with higher sensor densities. Table 3, derived from the works of (Patwari, Ash, Kyperountas, III, Moses, & Correal, 2005), (Gustafsson & Gunnarsson, 2005) and (Liu, Darabi, Banerjee, & Liu, 2007), summarizes the positives and negatives of the various approaches.

**Table 3 - Summary of Signaling Techniques**

| Method | Pros | Cons |
|---|---|---|
| **AOA** | +Used by cell towers to locate users<br>+Can be used for acoustical localization<br>+Provides direction to neighboring sensor | -Requires antenna array<br>-Potential accuracy around 10m<br>-Directional accuracy 5-10degs |
| **TOA** | +UWB demonstrated accuracy between 0.12 to 1.5m<br>+Clock scheduling specified in IEEE 802 docs | -Additive noise<br>-Multipath<br>-Other than UWB, accuracy is 5-100m |
| **TDOA** | +Can be used for acoustical measurements<br>+High resolutions clocks (GPS) yield very good accuracy | -Dependent on hardware chip rate<br>-Accuracy usually 5-50m |
| **RSS** | +Uses common infrastructure<br>+Calculation done in hardware<br>+WLAN RSS accuracy 2m | -Multipath and shadowing<br>-Transmission weakens when battery depletes<br>-Relies on a good fingerprint database |

The localization problem can be broken down into three different scenarios: with stationary beacons, with moving beacons, and beacon-free. Beacons are defined as nodes that are aware of their location either through hard coding or with a GPS. The nodes that are unaware of their position are called unknowns. When nodes are able to determine their location through various techniques, they can become beacons and relay location information to other unknown nodes. One of the obvious shortcomings of stationary beacons is the lack of robustness. If the nodes are going to be operating in various locations, particularly various indoor locations, the ability to setup new stationary beacons would be very time and resource intensive. In localization with moving beacons, the node has the ability to always know its exact position and by using range estimates can estimate the locations of the nodes it passes within range. While this is more feasible than using stationary beacons in different locations, knowing precise location indoors is very difficult. Beacon-free localization involves the removal of nodes that know exactly where they are and uses a coordinate system. The nodes communicate with one another and decide their distances based on signal estimations. To determine the exact locations of the nodes typically requires post-processing to translate the coordinate assignment to an absolute location or fast processing capabilities on the nodes themselves to handle the location translation.  A hybrid solution could be implemented between mobile beacons and a beacon-less system where a node that had recent location information, (i.e., goes from using GPS outdoors to going indoors) could share that information with the other nodes that do not have updated location information.

The July 2005 IEEE Signal Processing Magazine special edition on positioning and navigation contains excellent overviews on various techniques and algorithms. In (Sun, Chen, Guo, & Liu, 2005) positioning algorithms for WLAN are discussed. When comparing empirical models versus propagation models, empirical models place a client at a number of sample reference points and measure the RSS over several seconds. The propagation model is based on radio wave characteristics while it travels through a certain environment. The changes in an indoor environment make the empirical model more difficult to deal with as objects move and people may come and go as discussed in (Kaemarungsi & Krishnamurthy, 2004), among other sources. The propagation model can be affected by environmental changes like humidity which can alter the effectiveness of the signal propagation model. Sun et al. describe the various positioning algorithms as:

> "distributed (every node should be able to estimate its own location), the localized (each node gathers information from other nodes in its immediate neighborhood), the asymptotic convergence design (computation stops when a certain degree of accuracy has been achieved), the self-organizing scheme (node functioning does not depend on the global infrastructure), the robust design (the algorithm can tolerate node failures and range errors), and the cost-effective and energy-efficient approach (this algorithm requires little computation overhead)" (Sun, Chen, Guo, & Liu, 2005).

For the system related to this research, a combination of all algorithms could be used. This system is designed to be energy-efficient due to the battery life restrictions of smart phones. Also each node is able to determine its location while aiding others. It can also assume a certain level of accuracy for the sake of efficiency that is, once a location is deemed accurate to a specific threshold, the algorithm stops refining the position any further, thus saving valuable system resources. Another design consideration that (Sun, Chen, Guo, & Liu, 2005) raises are incremental versus concurrent algorithms. Incremental algorithms start with a few nodes that are aware of their location then add more via triangulation or local optimization schemes. Concurrent algorithms involve all nodes calculating their location estimation at the same time. Since incremental algorithms can propagate error more easily, concurrent algorithms can avoid this by continuously reducing errors among the nodes and is the focus of this research.

## 2.4 RSS and Fingerprinting

The mean received power at distance *d*, PL(d) is:

$$\boldsymbol{\overline{PL}(d) = P_0 - 10 n_p log \frac{d}{d_0}}, \qquad (1)$$

where $P_0$ is the received power in dBm at a short reference distance $d_0$, and $n_p$ is the path-loss exponent. Table 4 shows the various path-loss exponents for a given environment, particularly in free space, $n_p$ is 2, and in-building line-of-sight $n_p$ ranges from 1.6 to 1.8. Work done in (Tummala, 2005) suggests that for WLAN signals, $n_p$ ranges from 1.26 to 1.3 as tested between two different receivers and various rooms and buildings. Finding the correct value for $n_p$ can correlate the power received at a specific distance even if true data is unavailable.

**Table 4 - Path Loss Exponents for Different Environments (Rappaport, 2002)**

| Environment | Path Loss Exponent, n |
|---|---|
| **Free Space** | 2 |
| **Urban area cellular radio** | 2.7 to 3.5 |
| **Shadowed urban cellular radio** | 3 to 5 |
| **In building line-of-sight** | 1.6 to 1.8 |
| **Obstructed in building** | 4 to 6 |
| **Obstructed in factories** | 2 to 3 |

The equation, however, does not account for the surrounding environment clutter which can vary by the average predicted by the equation. Previous research has shown that the path-loss for a value *d* is random and distributed log-normally about the mean distance-dependent value. This results in a Gaussian distributed random variable that describes the transmit-to-receiver separation with clutter along the path as a random effect. The new equation is below with $X_\sigma$ being the zero-mean Gaussian distributed random variable in dB and standard deviation σ (Rappaport, 2002).

$$\overline{PL}(d) = P_0 - 10n_p log\frac{d}{d_0} + X_\sigma \qquad (2)$$

The transmitter-receiver distance is given by the Euclidean distance equation

$$d_{t,r} = \sqrt{(x_t - x_r)^2 + (y_t - y_r)^2} \qquad (3)$$

Revisiting algorithm possibilities, there are a few that deal specifically with RSS measurements. Work done by (Chandrasekaran, et al., 2009) reduced localization error to 0.24m and achieved very accurate results by comparing lateration based algorithms and classification based algorithms. The former explicitly models the signal-to-distance effect on RSS and estimates the position of the transmitter by measuring the distance to multiple receivers. The latter is also known as matching or fingerprinting algorithms and do not rely on a model of signal strength and distance relationship. They match RSS observations against an existing signal map.  Lateration-based algorithms generally take advantage of triangulation which uses the geometric properties of triangles to estimate the target location. An example of triangulation is seen in Figure 3 where the ideal intersection of three or more circles designates the nodes position.

Work done in (Liu, Darabi, Banerjee, & Liu, 2007) and (Gu, Lo, & Niemegeers, 2009) summarize the current state of cooperative localization for wireless networks. Some of the systems examined for their research along with some commercial systems are listed in Table 5.

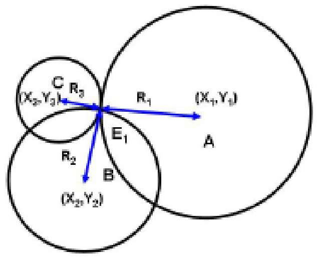


**Figure 3 - Triangulation typically used in cooperative localization when multiple access points are available**

**Table 5 - Summation of Techniques and Error Ranges**

| System | Wireless Technologies | Positioning Algorithm | Accuracy | Precision | Complexity |
|---|---|---|---|---|---|
| **Microsoft RADAR** [1] | WLAN, RSS | *K*NN, Viterbi-like algorithm | 3-5m | 50% within around 2.5m | Moderate |
| **M1**[2] | | Lateration (Bayesian inference) | 5.49m | | |
| **Horus** [2] | WLAN RSS | Probabilistic method | 2m | 90% within 2.1m | Moderate |
| **Ekahau** [2] | WLAN RSSI | Probabilistic method | 1m | 50% within 2m | Moderate |
| **SnapTrack** [2] | Assisted GPS, TDOA | | 5m-50m | 50% within 25m | High |
| **Ubisense** [2] | Unidirectional UWB, TDOA+AOA | Least Square | 15cm | 99% within 0.3m | Real time response |
| **Multi-Loc** [2] | WLAN RSS | SMP | 2.7m | 50% within 2.7m | Low |
| **GSM Finger-printing**[3] | GSM cellular radio | Weighted *K*NN | 5m | 80% within 10m | Medium |
| **CERP** [3] | GSM, FM, DVB, RSS | Discriminative gains | 1.37m | Median error of 0.16 | High |
| **DMRF** [3] | GSM, FM, DVB, RSS | Transformed-kernel | 1.48m | Median error of 0.33 | High |
| **SELFLOC** [4] | WLAN RSS and Bluetooth | Selective sensor fusion | 1.6m | Median error of 1.8 | Medium |

[1] (Bahl & Padmanabhan, 2002), [2] (Liu, Darabi, Banerjee, & Liu, 2007), [3] (Fang & Lin, 2010), [4] (Gwon, Jain, & Kawahara, 2004)

### *2.4.1 Lateration Based Algorithms*

Non-linear Least Square (NLS) algorithms estimate the true location of the transmitter (x,y) as an optimization problem where the actual locations of the reference points ($x_i$,$y_i$) are known beforehand. The distance estimate is obtained from the signal-to-distance relationship. They then solve for the optimal set that minimizes the sum (Chandrasekaran, et al., 2009) by

$$(\hat{x},\hat{y}) = \arg\min_{x,y} \sum_{i=1}^{N} \left[\sqrt{(x_i - x)^2 + (y_i - y)^2} - d_i\right]^2 \qquad (4)$$

Using this method, (Chandrasekaran, et al., 2009) achieved a median accuracy of 1.62m.

Baysian Networks (M1) encode dependencies and relationships among a set of random variables. The relationship between RSS and the location is found using the log-distance propagation model. This algorithm enabled (Chandrasekaran, et al., 2009) to achieve 0.24m accuracy. In (Patwari, Ash, Kyperountas, III, Moses, & Correal, 2005), they classify this type of estimation under distributed algorithms because the node likely does not have the processing power to perform the calculations. The work must be distributed because a single processor may cause a communication bottleneck. They note that this method is particularly promising because each sensor stores a conditional density on its own coordinates based on the measurements along with the conditional density of its neighbors.

### *2.4.2 Classification Based Algorithms*

The RADAR system proposed by (Bahl & Padmanabhan, 2002) uses a signal map as an input during an offline phase. During the online phase, a signal is matched with the closest fingerprint in the database. The method the Microsoft researchers used was K-Nearest Neighbor averaging where the algorithm searches for K location entries from the database which have the smallest root mean square of error. The coordinates returned are averaged to compute the final location estimate. In their experiment, the authors achieved a median error of 2.93m using three access points.

Gridded-RADAR is an improvised RADAR system in which the measurement area is sub-divided into a grid and the signal map is interpolated over the entire grid. Work from (Chandrasekaran, et al., 2009) found that this provides a much finer-grained resolution because the regions not covered by the signal map can be returned as location estimates. In their experiment, the median location error was 0.36m versus the 2.93m found from the original RADAR system.

The Highest Probability (H1) algorithm divides an area into tiles and returns the most likely (*x*,*y*) position by finding the highest probable tile using Bayes' rule over the set of RSS values. This approach assumes that the distribution follows a Gaussian distribution. More detailed information about the H1 algorithm is in (Chandrasekaran, et al., 2009).

## 2.5 Data Fusion

Data fusion is the process of dealing with the association, correlation, and combination of data and information from single and multiple sources. The goal is to refined position estimation more so than any single source could do. From (Sun, Chen, Guo, & Liu, 2005), "different sources, however, are subject to different propagation errors that contribute unequally to global position estimation errors. Adaptive data fusion and hybrid localization techniques are employed to better integrate different types of position and navigation information." For instance, GPS and cellular positioning combined achieve a much better result than either could independently. Both (Sun, Chen, Guo, & Liu, 2005) and (Sayed, Tarighat, & Khajehnouri, 2005) from the July 2005 IEEE Signal Processing Magazine highlight that data fusion techniques have been applied to TOA-TDOA, TOA-AOA, and AOA-TDOA measurements each with their own merits. For example, AOA and TDOA measurements have been combined to limit multipath effects. As the number of data sources grows, the process to control how they are managed and integrated into the system must be capable of handling the sporadic nature of signal processing.

One of the more familiar methods for data fusion is the Joint Directors of Laboratories (JDL) data fusion model described in (Hall & Llinas, 1997), (Llinas, Bowman, Rogova, Steinberg, Waltz, & White, 2004) and (Steinberg & Bowman, 2009). Some of the key issues they raise are: what algorithms are appropriate? What accuracy can realistically be achieved by data fusion? How does the environment affect the processing? Under what conditions does multisensory data fusion improve system operation? These are all important to balance particularly when trying to develop a real-time system for cooperative localization. The model describes five levels:

- L0 - signal/feature assessment – Estimation of signal or feature states
- L1 - entity assessment – Estimation of entity parametric and attributive states
- L2 - situation assessment – Estimation of the structures of parts of reality

- L3 - impact assessment – Estimation of the utility/cost of signal, entity, or situation states, including predicted utility/cost given a systems alternative courses of actions
- L4 - process refinement – A system's self-estimation of its performance as compared to desired states and measures of effectiveness

As shown in Figure 4, these levels are not hierarchical and any signal source can be in any level independent of another. Level 0 is concerned about the structure of the measurement sets and not their cause. Here is where features of the signal are extracted to produce estimated signal and feature states and a level of confidence. In the terms of this effort, it requires differentiating between the various signals, and establishing a beginning weight value either though a user-input system, or based on previous use of the signal. Level 1 was conceived as dealing with highly-developed applications of data fusion such as detection, identification, location and tracking of physical objects. This idea of target detection and identification does not apply directly to cooperative localization in the sense that we are not tracking an object, but are listing the object that needs to be located. For the purpose of this assessment, level 1 is omitted from the proposed model. Level 2 is about inferring situations, as the name implies. If applying the model to system states, this level infers from the estimated state of one entity in a situation to another and from the estimated attributes and relationships of entities to situations. One of the difficulties with cooperative localization is that research will deal strictly with indoor environments or with outdoor environments but not precise measurement (i.e., GPS). The situation assessment level would aid greatly in determining what signal propagation parameters to use, either low np for outdoors, or high np for indoors. Another application could be when using a fingerprinting database, using one calibrated for indoors, or one for outdoors. Level 3, impact assessment, was meant primarily for tactical military decisions in the sense of if we follow this course of action, then the outcome could be that. Similarly, level 3 could be used in the cooperative localization sense of if we weigh this sensor more heavily; the precision will be affected in this manner. Finally, level 4 encompasses assessment, adaptive control, and data collection of the fusion process. This could, and should, include measures of performance and measures of effectiveness. The author's in [23] describe a proposed level 5 for user refinement. This level would be the transformation of the signal data into a graphical display or control board. While this level would be beneficial, it could also be tied into the far right of Figure 2 as the computer interface. An interesting contribution that [23] makes is that of including resource management levels that mirror the data fusion ones. Describing and incorporating them for this research is out of the scope of this paper, but they could be used for dynamically changing the level characteristics and parameters of the data fusion model.

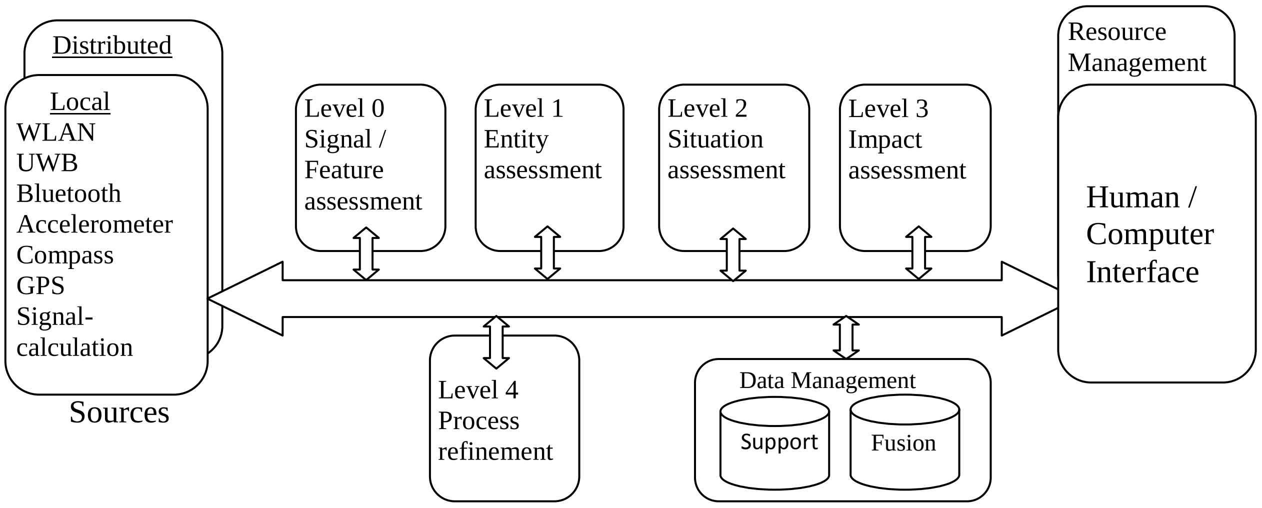

Distributed
Local
WLAN
UWB
Bluetooth
Accelerometer
Compass
GPS
Signal-calculation
Sources
Level 0 Signal / Feature assessment
Level 1 Entity assessment
Level 2 Situation assessment
Level 3 Impact assessment
Resource Management
Human / Computer Interface
Level 4 Process refinement
Data Management
Support
Fusion

**Figure 4 - JDL Data Fusion model, slightly revised for cooperative localization**

## 2.6 Improving Accuracy

A promising approach to achieving a much more energy efficient and resource limited location estimate is to use the concept of sensor fusion. In this approach, traditional cooperative localization techniques are combined with other available sensors. Cooperative localizing is handy using a smart phone as the localization device because many sensors (as discussed in section 2.1) are available. The approach has been done on other devices that combine the following: WLAN and Bluetooth (Gwon, Jain, & Kawahara, 2004), WLAN and accelerometer (Hamilton, Ma, Baxley, & Walkenhorst, 2010) (Xu, Ouyang, Le, Ford, & Makedon, 2007), or WLAN and other RF signals of opportunity (Fang & Lin, 2010).

### *2.6.1 WLAN and Bluetooth*

The more widely referenced article, written in 2004, (Gwon, Jain, & Kawahara, 2004) proposes the Selective Fusion Location Estimation (SELFLOC) and Region of Confidence (RoC) algorithms. The former infers the user location by selectively fusing location information from multiple wireless technologies while the latter attempts to overcome the problem of aliasing in the signal domain. Their SELFLOC system allows them to combine triangulation, KNN and Smallest M-vertex Polygon (SMP). SMP, which has been used in (Pandya, Jain, & Lupu, 2003) involves searching M candidates from each access point whose distance in the signal space matches in order to create m-vertex polygons. The coordinates of the smallest polygon are averaged to give the final location. The SELFLOC approach combines each algorithm with an average weight based on certain confidence factors (Figure 5). The RoC algorithm they propose counters aliasing in the signal domain by forming regions of confidence within which the true location of a user lies with some high probability.

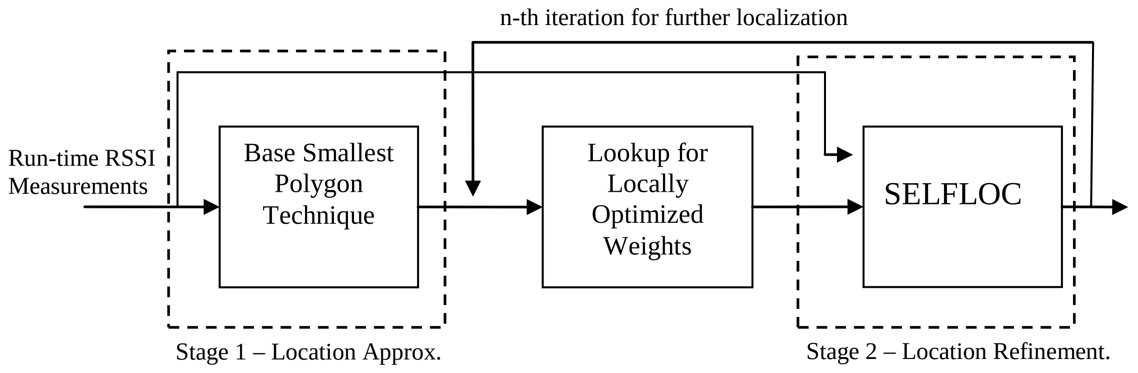

n-th iteration for further localization
Run-time RSSI Measurements
Base Smallest Polygon Technique
Lookup for Locally Optimized Weights
SELFLOC
Stage 1 – Location Approx.
Stage 2 – Location Refinement.

**Figure 5 - SELFLOC system (Gwon, Jain, & Kawahara, 2004)**

Various findings were found by comparing WLAN, Bluetooth, number of AP's, and algorithms. In general, when using both Bluetooth and WLAN, accuracy increased by 11-28% and when all algorithms were used improvements rose to 47-70%. While impressive improvements were noticed with SELFLOC and RoC, the process requires extensive computation requirements and would have to be tailored appropriately for the smart phone environment.

### *2.6.2 WLAN and Accelerometer*

In (Hamilton, Ma, Baxley, & Walkenhorst, 2010), they combine the RSS measurements with accelerometer results to overcome inaccuracies with RSS measurements alone. They apply a distributed extended Kalman filter to combine the two measurements in their simulations. Their simulation includes anchor nodes. They note that acceleration is a continuous process but they sample at specific intervals potentially missing valuable information from the sensor. Therefore, their measurement is used as the average acceleration over the entire interval between checks. In their simulation of 60 mobile nodes, their hybrid approach was able to use the acceleration measurements to generate higher accuracy in location estimation despite the low distance measurement accuracy from RSS measurements.

Another effort that used accelerometers is detailed in (Xu, Ouyang, Le, Ford, & Makedon, 2007). They propose an Anchor-Free Mobile Geographic Distribution Localization (MGDL) algorithm. The use of an accelerometer is less involved than in (Hamilton, Ma, Baxley, & Walkenhorst, 2010) and instead relies on the sensor to determine when the node resumes movement. They perform their algorithm by labeling nodes as either static or mobile and updated or not. Since they assume all nodes will start as static and not updated, they perform something similar to hop-counting to measure the distance from some bootstrap node to other nodes. Each node collects coordinate information from its neighbor and then Dijkstra's algorithm is performed to obtain the shortest path between each pair. A local map is then constructed followed by a global map. The nodes are now labeled as static and updated. At this point, the accelerometer is used to detect when movement occurs so that the nodes location can be updated. When compared to other algorithms such as Monte Carlo Localization (MCL) and Elastic Localization Algorithm (ELA), MGDL outperformed both when the number of nodes increased (20% better location accuracy than MCL and 28% better than ELA), when the node speed varied (22% better location accuracy than MCL), and that communication overhead is better in MGDL while MCL maintains a constant overhead.

### *2.6.3 WLAN and Signals of Opportunity*

Two algorithms that (Fang & Lin, 2010) propose are Direct Multi-Radio Fusion (DMRF) which uses the spatial correlation after the information of measurements is reorganized to minimize redundancy and Cooperative Eigen-Radio Positioning (CERP) which incorporates the spatial discrimination to estimate the location. The DMRF algorithm assumes each node can detect RSS values from GSM (cellular network), Digital Video Broadcasting (DVB) and FM (radio). They use kernel positioning instead of the more common Gaussian-based method. A probabilistic function is estimated by non-linear calculations of the transformed kernel distance between the joint observation and all stored RSS patterns. The authors conduct an experiment using both algorithms using GPS as ground truth data. Comparing the three signals separately, then combine along with SELFLOC found that CERP performs the best with a mean error of 1.37m followed by DMRF at 1.48m, then SELFLOC at 2.68m, WLAN at 2.69m and finally GSM with an error of 8.43m.

## 2.7 Summary

This chapter presents background information on the Android operating system and cooperative localization. The technique of RSS fingerprinting is discussed along with a few algorithms that are implemented to optimize the fingerprint matching such as lateration based and classification.  Finally, the chapter concludes with how accuracy improvements are made by incorporating various sensors. In all instances, adding an additional sensor improved location estimation.

# III. Methodology

## 3.1 Chapter Overview

This chapter defines the methodology for evaluating cooperative localization on constrained systems. Section 3.2 describes the problem definition along with the goals and hypothesis of this work. Section 3.3 identifies the experimental setup, boundaries and assumptions of this system. Section 3.4 offers the system services while section 3.5 discusses solution workload. Section 3.6 identifies the performance metrics for determining the merit of this work. Section 3.7 defines key workload and system parameters while section 3.7 and 3.8 discuss the evaluation technique and experimental design. Finally, section 3.10 serves as a summary to this chapter.

## 3.2 Problem Definition

The problem considered in this research is how nodes collude to share their location information in contested environments. Current combat operations and search and rescue missions take place in urban settings where knowing the location of another friendly user can mean the difference between life and death.  Since many people already carry smart phones, they can use its numerous sensors to aid in location estimation.

### *3.2.1 Goals and Hypothesis*

The objective of this research is to bring cooperative localization to computationally restricted devices, primarily the Android platform, in order to utilize several sensors that are now available on these common devices. Most of the research that has been done in the cooperative localization field has been on specialized equipment with ideal laboratory conditions. The few works that incorporate multiple sensors and techniques have done so in simulation or have relied on post processing of the data. Another limitation of most research is that the location estimation is only as good as the fingerprinting database built *a priori*. If the time is not taken to ensure a good database, or the operating environment changes, the results of the location estimation are less than optimal. By using common devices, this research demonstrates that cooperative localization can be achieved with low-cost, off-the-shelf, multiple sensor devices.

The goals of this research are to:

- Bring real-time cooperative localization to computationally constrained devices by using their Wi-Fi hotspot capability, Bluetooth, accelerometer and other sensors available to the device.
- Organize the data fusion with guidance from fusion methods not previously applied to cooperative localization.
- Demonstrate this ability on commercial off the shelf hardware and examine strengths and weaknesses of the hardware.

The goals are realized by producing a proof-of-concept application that can indicate neighbor nodes with improved accuracy over methods that do not utilize the various sensors. It is hypothesized that by utilizing the many sensors on the phone that the device will be able to update more reliably and in near real time to produce more accurate location estimates.

### *3.2.2 Approach*

The approach to developing the software for the system will follow the spiral model. This combines both top-down and bottom-up software engineering concepts which allows for designing and prototyping to occur rapidly. This approach focuses the effort by keeping objectives in mind, resolving risks, developing and testing code, and planning for the next sensor to be implemented into the project.

## 3.3 System Setup

The system includes the Android devices within a wireless sensor network. The devices can receive, transmit, and process data. For simplicity of drawing, the transmitting and receiving range are assumed to be symmetrical but in the real world, the distances may vary due to changing conditions. While the RF waves continue to propagate indefinitely, there is a limit to the minimum amount of power the node needs in order to use the signal for RSS calculation. This is handled by the hardware and software already built into the Android operating system. Since the research goal is to use a COTS system with multiple sensors to localize in the real world, the physical network layer is considered in the experiment and error free communication cannot be assumed. Error free communication can be attempted by keeping the distance relatively short. This communication functionality along with the medium access control (MAC) protocol is handled by the Android kernel. The nodes may or may not be stationary. A moving node allows for the positioning algorithm to update and correct for any errors that may have been encountered. All of the devices use the same positioning algorithm.

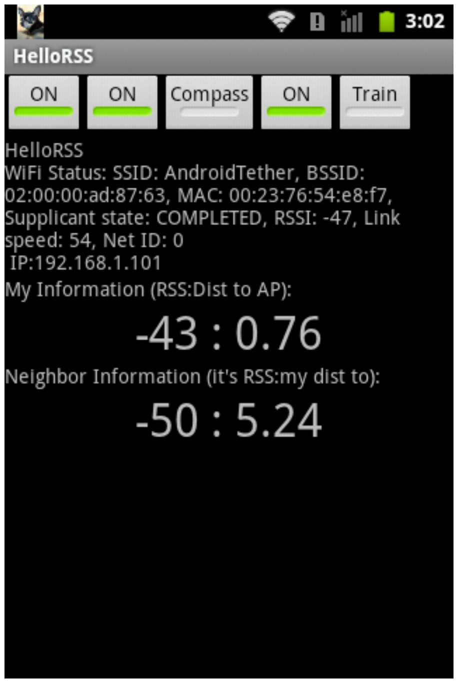
As mentioned in Chapter 2, the phones need to be rooted and upgraded in order to utilize the Wi-Fi Hotspot capability. In order to do that on the HTC phones, the programs `ClockworkMod Recovery` (Dutta, 2011), `Universal Androot` and a flash image file were used as described on several Android forums (Cyanogen, 2011). `ClockwordMod` is a tool that allows the user to perform several advanced recovery and installation operations on the phone. These are normally locked from the typical user but by installing the program, new operating systems can be put onto the device. `Universal Androot` is a program that is installed on the vendor version of the operating system and when executed exploits a common vulnerability to give the user root level access. When root access is achieved, the flash image is placed on the phone's memory card and the command "`flash_image recovery recovery-clockworkmod-hero.img`" is invoked to put the recovery image onto the system partition. This flash image is a small utility that allows the user to rewrite system partitions with image files, such as the `ClockworkMod` program to simplify installing a modified operating system. The process is similar for the Motorola handset, with a different flash image and version of `ClockworkMod`. With `ClockworkMod` working successfully, as observed by rebooting the system into the bootloader, `CyanogenMod 7` can be installed.

Both devices are capable of running `CyanogenMod 7`, which is an aftermarket version of Android 2.3.7 that offers features not found in official handset releases (CyanogenMod, 2011). In other words, when a handset company stops releasing updates provided by Google, `CyanogenMod` will continue supporting it with the latest version and provide some of the functionality that the original developer omitted. Installing this operating system required putting the designated zip file on the memory card and booting into `ClockworkMod`. From there, a factory reset and cache wipe are completed followed by the option "`Install zip from sdcard`". `CyanogenMod 7` is installed and the devices are ready to be used. In order to create a personal network for this experiment, the open source application called `android-wifi-tether` (Mue, 2011) is installed. This program allows users to create a Wi-Fi hotspot and while most users install it to share their phones internet capability, for this research it is used to create a subnet for each node to broadcast location updates. At this stage, the programming portion can begin.

The Windows version of the Android Software Development Kit (SDK) is used to implement the localization method. The target application uses API Level 8 which corresponds to the operating system version of Android 2.2, revision 3 (July 2011). A software database stores RSS measurements and their corresponding distances from experimentation. When a device can connect and exchange location estimates and relevant information about the network, more sensors can be incorporated. In other words, when node *a* can connect to node *b*'s Wi-Fi hotspot and node *b* transmits its location estimate, node *a* uses the estimate with the related RSS measurement in the fingerprint database. Once this capability is achieved, another measurement approach can be introduced into the system. One of the next logical pieces of software to include in the next phase is the use of the accelerometer to detect when the device moves. If it is currently moving, the magnetometer can be polled to use the compass feature in the API. By knowing the heading of the device, further refinements to fluctuating RSS measurements are made. This process allows the fingerprint database to be updated, since the node knows which direction it is heading, it can increase or decrease the correlating distance appropriately.

As mentioned previously, the RSS database is an important aspect for localization indoors. The SQLite database is created by collecting RSS values every 3 seconds over the span of 2 minutes per distance. Each measurement is every 0.254m from the wireless access point. This corresponds to about 40 measurements per location. The average are taken and recorded. When 30 locations are measured, the averages are plotted to characterize the path-loss curve described in Chapter 2. A logarithmic trend line is fit to the measured curve to create a one-to-one mapping of RSS levels and distance. The degree of error can be estimated by subtracting the distance from the measured recording and the distance that corresponds to the same number from the modeled trend line. For example, a measured RSS of -66dB corresponds to 1.524m, but the model has -66dB at 1.397m resulting in an error of 0.127m.

A Unified Modeling Language (UML) diagram for this research is included in Appendix A. The UML diagram shows each Android class with relevant variables and functions along with the interclass communication. It can be referenced for the remainder of this section. The user interface has buttons for the user to select what service they wish to run: receive, transmit, compass and accelerometer and is shown in Figure 6. These buttons control the on/off functionality of each service by creating and sending an intent for the service to receive. The screen also displays the current Wi-Fi configuration such as IP address, initial RSS reading, and to what access point it is connected. This main screen also updates with the phone’s current RSS reading, distance to the access point, the other node’s RSS reading, and the distance between the two. The activity must have a broadcast receiver from the application program which broadcasts the information as necessary.



HelloRSS
WiFi Status: SSID: AndroidTether, BSSID: 02:00:00:ad:87:63, MAC: 00:23:76:54:e8:f7, Supplicant state: COMPLETED, RSSI: -47, Link speed: 54, Net ID: 0
IP:192.168.1.101
My Information (RSS:Dist to AP):
-43 : 0.76
Neighbor Information (it's RSS:my dist to):
-50 : 5.24

**Figure 6 - Screenshot of localization application**

The transmit service is responsible for encapsulating information and transmitting it via a User Datagram Protocol (UDP). This allows the device to send messages to other nodes without handshaking and prior communication. This method may be unreliable as packets are not guaranteed to be delivered, but for this small network and the frequency of message transmission, this is not a concern. The transmit service also requests location updates from the GPS and receives broadcast intents from the accelerometer and compass. This information is transmitted as a UDP packet at a regular rate to be received by the receive service. Implemented in a handler, which allows messages to be sent and received from the operating system, the service creates a socket to wait for UDP messages to arrive. When a new transmission is received, it is parsed for what type of packet it is, such as a fresh GPS, stale GPS, movement, or stale movement packet.

In order to efficiently transmit the data, the packet is formed with relevant information. The packet starts with an identifier, such as ‘gps’, ‘sgps’, ‘moved’, ‘smoved’, or ‘est’. These packets represent active GPS, stale GPS, moved, old moved, or estimated location data respectively. All but the ‘moved’ and ‘smoved’ packets send string representations of the `Location` class information such as latitude, longitude, altitude, bearing, accuracy, speed and time in that order. Finally, the transmitting node’s RSS value is added. For example, the first transmission from a node sending GPS data would be:

gps,39.781127,-84.08111622,0,90,0,0,1325681676393,-42

where the field and corresponding values are shown in Table 6.

**Table 6 - Example GPS transmission**

| Item | Value |
|---|---|
| **Type** | GPS |
| **Latitude** | 39.781127 |
| **Longitude** | -84.08111622 |
| **Altitude** | 0 |
| **Bearing** | 90 |
| **Accuracy** | 0 |
| **Speed** | 0 |
| **Time** | 1325681676393 |
| **RSS** | -42 |

A moved packet would contain the X and Y values from the `SensorEventListener` along with the bearing and RSS value. The stale movement packet only needs to send the timestamp of the last move along with the bearing and RSS value. By knowing the time the node last moved and its RSS value, a better approximation can be made of its location. Examples of a move and stale move packet are below:

Moved,-3.2961242,-0.19068487,0.0,-42

Smoved,1325682747867,0.0,-46

Depending on the type of packet, a specific function is called in the Application class.

The Application class is the main file responsible for computing the location estimates. For instance, a fresh GPS transmission has the accuracy checked as reported by the `Location` class accessor `getAccuracy()` to determine if the returned integer is more accurate than what was previously stored in the database for that node. By contrast, the stale GPS information is weighed less heavily as an accurate location for that node or a movement packet would add some distance to the estimated distance between the node and access point based on how fast the movement was. This class also provides a way for the Activity previously mentioned to display the distance information, along with accessors so that other classes can easily get the location of their node and the other node.

The compass and accelerometer services follow similar designs. `SensorEventListeners` are created which require the implementation of the function `onSensorChanged(SensorEvent event)`. The event argument that is passed through for the accelerometer is a structure that includes accuracy, type of sensor, timestamp, and values. The `values` element is an array of floats that characterize the X, Y and Z axis which correspond to left/right, forward/backward, and up/down movement, respectively. These values include the rate of gravity, so a device sitting flat on the table responds with a value around 9.8 in the positive z-direction. This research restricts movements to the X and Y axes, since the nodes will only be on a 2-D plane. Restricting the experiment to two dimensions reduces the uncertainties with the phone’s antenna pattern. When creating a sensor of `type_orientation` for the compass, the event values correspond to azimuth, pitch and roll. The only direction used in this effort is the azimuth, to determine which way the phone is pointing, because the phone is assumed to be lying flat thus canceling out pitch and roll. In order to simplify the positioning algorithms, the hotspot is assumed to be facing north so that a latitude and longitude can be calculated given the access points latitude and longitude, distance, and azimuth to the node. Both the accelerometer and compass service broadcast the event values to the main application.

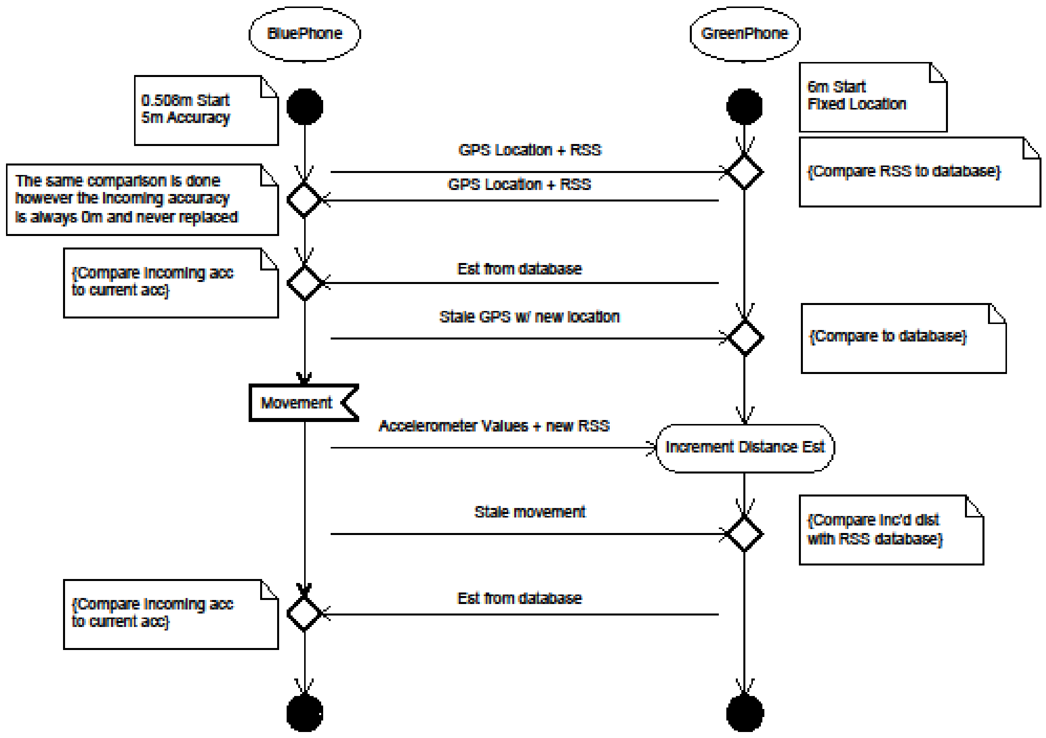
## 3.4 System Services

This system enables an Android smart phone to estimate its position without the direct aid of GPS or cellular tower triangulation. The system achieves this by sending a minimal number of messages in a short period of time. It is successful when all nodes in the experiment within a given transmit area are able to estimate their location. The system fails when a node has not been able to determine its relative position to the other nodes. It is reasonable to assume that when a node is not in transmit distance of any other node that it has no ability to estimate its location and thus fail. Some other causes for failure could result from unreliable medium, such as the introduction of a barrier that prevents signal propagation or too many nodes operating on the same wireless frequency channel. This research does not introduce any barriers that are not already present at the time of testing. In other words, an office cubical that is already present is modeled, but a sheet of metal will not be placed in between two nodes during testing. This research also relies on proper IEEE 802.11 handling of network congestion.

## 3.5 Workload

The workload of the system depends on the network topology. For sparse networks, there is less network congestion; however, the localization results may be less accurate. When the network is dense with several nodes, the localization results are more accurate but the network congestion is considerably higher. This could result in more dropped packets. In order to exchange location information, nodes connect over IEEE 802.11 and follow standard network association procedures, which include authentication and association requests/responses with the node acting as the access point. This process of connecting and disconnecting from each node increases the amount of traffic in the network is not directly related to the localization process.

An example of the communication between two phones is illustrated in Figure 7. Both phones start with a location and accuracy. After GPS messages are exchanged, the phones send an estimate of where they think the other is. If the accuracy is better, the new location is accepted. When movement is detected, a moved packet is set with the accelerometer data for the other phone to approximate a new distance. Another comparison is done and another estimate packet is sent out to determine if the new estimate is worth keeping.



BluePhone
GreenPhone
0.506m Start
5m Accuracy
6m Start
Fixed Location
GPS Location + RSS
GPS Location + RSS
The same comparison is done however the incoming accuracy is always 0m and never replaced
{Compare RSS to database}
{Compare incoming acc to current acc}
Est from database
Stale GPS w/ new location
{Compare to database}
Movement
Accelerometer Values + new RSS
Increment Distance Est
Stale movement
{Compare Inc'd dist with RSS database}
{Compare incoming acc to current acc}
Est from database

**Figure 7 - Example communication between nodes**

## 3.6 Performance Metrics

The primary metric used to evaluate the performance of the system is the distance error. As discussed in Chapter 2, the Euclidean distance can be used to measure the distance between two nodes. In order to find the error, the real distance is subtracted from the computed distance. For example, given a transmitter, t, and a receiver, r, the real distance between them will be based on $(x_t, y_t)$ and $(x_r, y_r)$. The calculated distance will be based on $(x'_t, y'_t)$ and $(x'_r, y'_r)$. The distance error, $e_{t,r}$, is the difference between the real distance, $d_{t,r}$, and the calculated, $d'_{t,r}$ as shown in the following equation

$$\boldsymbol{e_{t,r}} = \left| \sqrt{(\boldsymbol{x_t} - \boldsymbol{x_r})^2 + (\boldsymbol{y_t} - \boldsymbol{y_r})^2} - \sqrt{(\boldsymbol{x'_t} - \boldsymbol{x'_r})^2 + (\boldsymbol{y'_t} - \boldsymbol{y'_r})^2} \right| \qquad (5)$$

This effort has been simplified for only one direction of movement, so the *y* component reduces to zero and the error is simply the estimated distance minus the actual distance.

In order to further characterize the system error, the maximum error is calculated. All efforts to localize a node contain some source of error. The error is calculated by taking the maximum error of the localized node and combining it with the error of the node that needs to be localized. In other words, if the localized node has an error of 2m and the computed distance between them is 5m with a 1m error at the newly localized node, the worst case distance between the two nodes is 8m. Based on general observations of the GPS capability on the HTC Hero, an error of 1.8m is common with 9 GPS satellites being used. This value is used in the analysis because determining the best and worst case values of the GPS chip is out of the scope of this research. By assuming an error of 1.8m for the worst case analysis, a general idea of how the system is inaccurate is plotted to show the effects of the error. The maximum error could obviously be higher with a larger GPS error but the trend is similar.

## 3.7 System Parameters

### *3.7.1 Workload Parameters*

- Number of Nodes – The number of nodes in the system can affect how accurately one is able to determine its location. This experiment consists of two localizing nodes.
- Node Density – Node density defines how many nodes are within transmission radius. If no nodes are within the transmission radius, the density is considered

negligible. If all available nodes are within the radius, then the density is maximal. While all of the nodes (2) are within the transmission radius, the density is still low compared to other research efforts. The low density requires the nodes to be more accurate in their location estimation and cannot rely on reducing error via many neighbors.

### ***3.7.2 System Parameters***

- Node Range – The node range is the distance the node can transmit and receive packets. As the battery depletes, the range will go down and the reliability of RSS measurements may be affected. Based on the transmission distance of a mobile access point, the range will not exceed 7m. The distance is controlled by measuring the distance and maintaining line-of-sight with the hotspot.
- Accuracy – The accuracy is how close the location estimate is to the actual distance. The node localized poorly if the accuracy is low (i.e., true distance was much higher or lower than the estimate).
- Sensors Used – The types of sensors used in the approximation aid in the merit calculation of the localizing algorithm. This relies on the accuracy; if the accuracy is high and a certain collection of sensors was used, they are favored in other rounds of location estimation.
- Antenna Type – The antennas used are capable of bi-directional links and omnidirectional transmission. It is assumed that any node that can receive a transmission also has the ability to transmit to that distance. This assumption reduces the likelihood of dropped packets causing nodes not to localize.

## 3.8 Evaluation Technique

Direct measurements are used as an evaluation technique for this research. Simulators often have difficulty accurately modeling the RF environment and make several assumptions. Therefore, producing a system that works in the real world provides the most realistic results. Also, characterizing the exact computational capabilities and the sensors available on an Android device would be more challenging in simulation than with real devices due to the fact that most modeling environments do not have nodes pre-written for Android devices. Allotting time to create new nodes detracts from working with the actual hardware.

Primary testing is indoors in an office building environment with concrete walls and cubicles to avoid weather limitations. Although this environment produces a more undesirable RF atmosphere than a wide-open outdoor space, it does not introduce weather uncertainties and provides a more realistic environment since the goal is for the system to work in contested spaces. Artificial location information loaded onto each node with a certain degree of accuracy. This would be analogous to a node coming from outside into the Wi-Fi hotspot. This hotspot is at one end of a tape measurer and the two nodes are placed at varying locations, as shown in Figure 8. The reported location from the hotspot from each node and the distance between the two nodes are noted along with the actual distances. This approach allows for a degree of error to be determined.

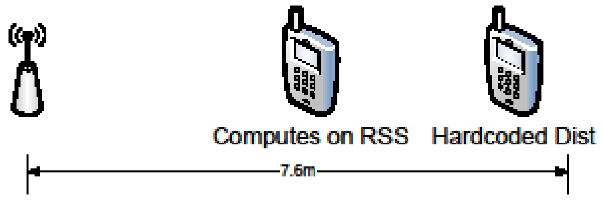


**Figure 8 - Simple testing diagram showing how experiment will be setup**

## 3.9 Experimental Design

When considering how many experimental runs to execute of a hardware experiment, practical considerations need to be taken into account. For instance, running experiments for every potential sensor combination are time intensive. This research bases the merit of sensor fusion on computationally constrained devices ground off recent works in the field with only RSS computation. The merit of this research relies on the error distance previously mentioned. In order to achieve a reliable estimate, the average error distance over 30 different samplings at the same distance is used. This number of trials provides a sample mean that is approximately normally distributed with a mean and variance, according to the Central Limit Theorem. More experiments can be done at varying distances and movements with the same number of runs to show how the system performs under various conditions. For simplicity, measurements will be taken at 0.254m, 1m, 2m, 3m, 4m, 5m, 6m, and 7m. The measurements alternate between pairs of distances, for instance, the phone starts at 0.254m and the estimate are recorded, and then moved to 1m, recorded, then back to 0.254m, and so on.

To test the accelerometer and compass, the phone is placed on a cubical desk with the respective service running for the test. The `SensorEvent` values are sent to the debugger for analysis. Using a hand, the phone is moved left to right, right to left and again left to right. The other test consists of movement up (positive *y* direction) followed by down (negative *y* direction). A third test is conducted to examine the values when the phone moves in a quarter circle motion. The motion is deliberate and precise over a distance of one foot. The compass is tested by keeping the device stationary and pointed in a known direction as proven by a secondary compass. To test the susceptibility to interference and  poor calibration, another phone is moved over top of the test phone.

## 3.10 Summary

By combining sensors in a methodical, useful manner, the ability to determine one's location in a wireless network should be more reliable. This research shows that common devices can be configured to deduce one's location in constrained environments effectively and more accurately than previous works demonstrate. This chapter outlines the goals and approach of this research and defines the system boundaries, assumptions, and parameters. An assessment of the evaluation technique is discussed along with the design of the experiment.

# IV. Analysis and Results

## 4.1 Chapter Overview

This chapter examines the results from the cooperative localization processes that are applied on the Android operating system. To differentiate between the two HTC phones, one contains blue wallpaper while the other has green wallpaper.  Other than the wallpaper, nothing differs between the phones. First, Section 4.2 details and analyzes the results of localizing nodes on Android as described in this effort. These results show that radial distances further from the access point tend to under estimate their distance while distances closer to the access point are over estimated.  Nodes in the middle of the distance range had a closer average estimate. Next, Section 4.3 explores the investigative questions answered from the experiment. Finally, Section 4.4 concludes the analysis and results chapter with a summary.

## 4.2 Results of Experiment Scenarios

While pursuing cooperative localization strictly on mobile devices, several observations are discovered through the development process. First, since the RSS database is used heavily for distance correlations, it is discussed in Section 4.2.1. Next, brief generalizations about the accelerometer and compass are mentioned in Sections 4.2.2 and 4.2.3 respectively. The outcome of the resulting system is explored in Section 4.2.4. Finally, a demonstration of how bad a few select results could be is shown in Section 4.2.5.

### *4.2.1 RSS Observations*

Attempts to create an accurate RSS fingerprinting database are executed several times in various locations. The first observation is conducted to determine the measuring capabilities of the system. This experiment did not follow the 2 minute averaging process described in Section 3.3 and instead focused on observing a debugging log that was added for troubleshooting purposes. The RSS value is printed to the Eclipse LogCat window and the rough average is recorded for each distance. Figure 9 shows the graph of RSS readings corresponding to distance. The graph follows a general logarithmic curve using a path-loss variable of 3.01.

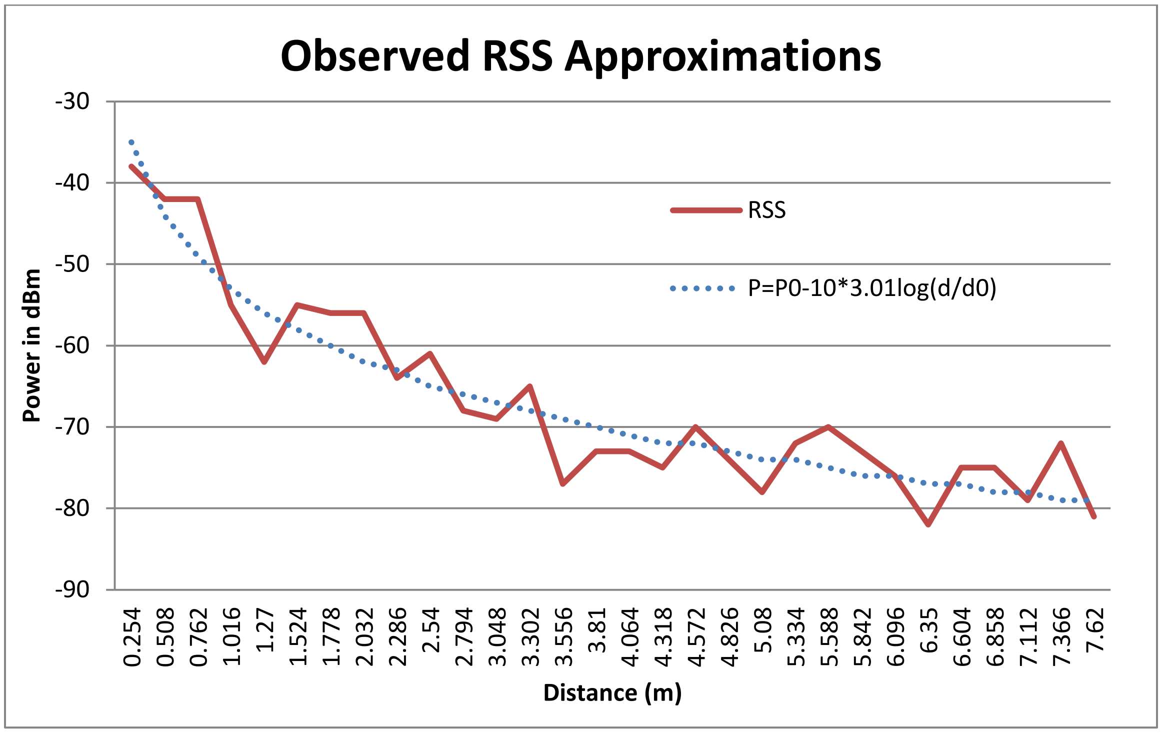


**Figure 9 - Observed RSS values in hallway**

In order to examine the RSS measurements more closely, a new experiment is conducted with a log of every RSS measured. Figure 10 shows the graph of every RSS value recorded over the span of 64 minutes, or roughly 1300 RSS measurements at 1 measurement every 3 seconds. Averaging those values every 2 minutes (the time at which the phone was moved to the next location) yields Figure 11. Once again, a logarithmic trend line is fitted and a corresponding path-loss line. In order to match the trend line, a path-loss value of 1.9 is chosen. The result is below the free-space value of 2 found in prior research but higher for the value expected for line of sight in a hallway with brick walls. If the data after approximately 4 meters is discarded, the resulting trend line (shown in dotted blue) more closely relates to the first half of the data with a path-loss variable of 3.2. Because data cannot simply be discarded, more measurements are taken with the two phones.

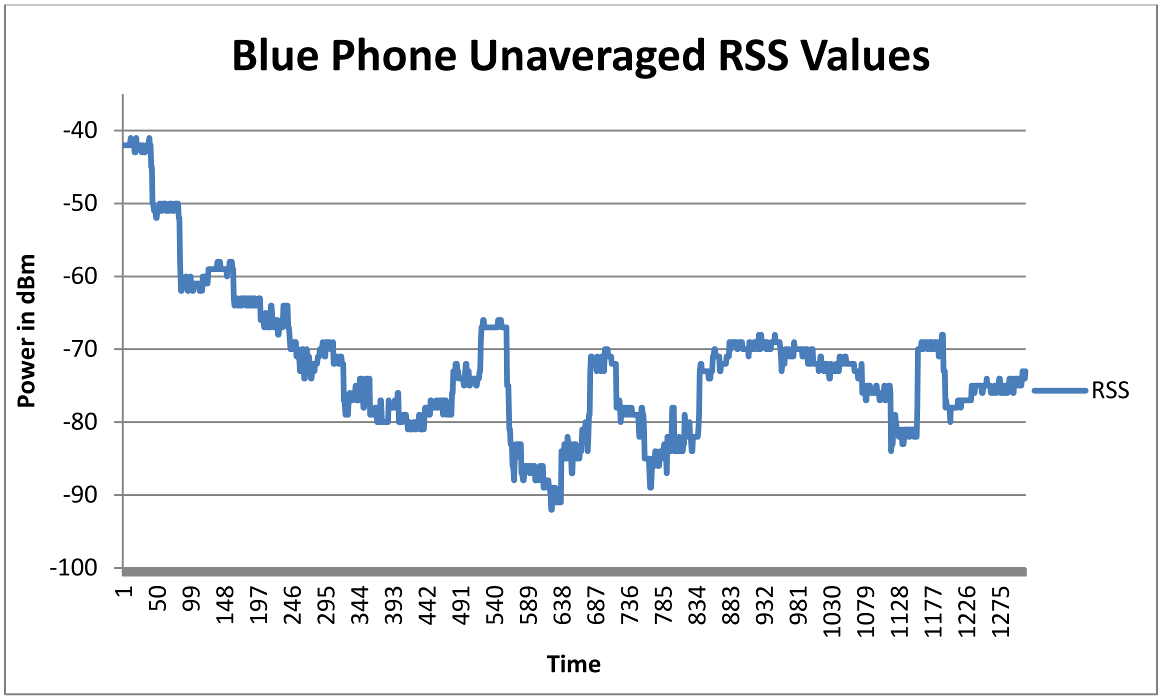


**Figure 10 - RSS recordings in hallway; 2 min at each location**

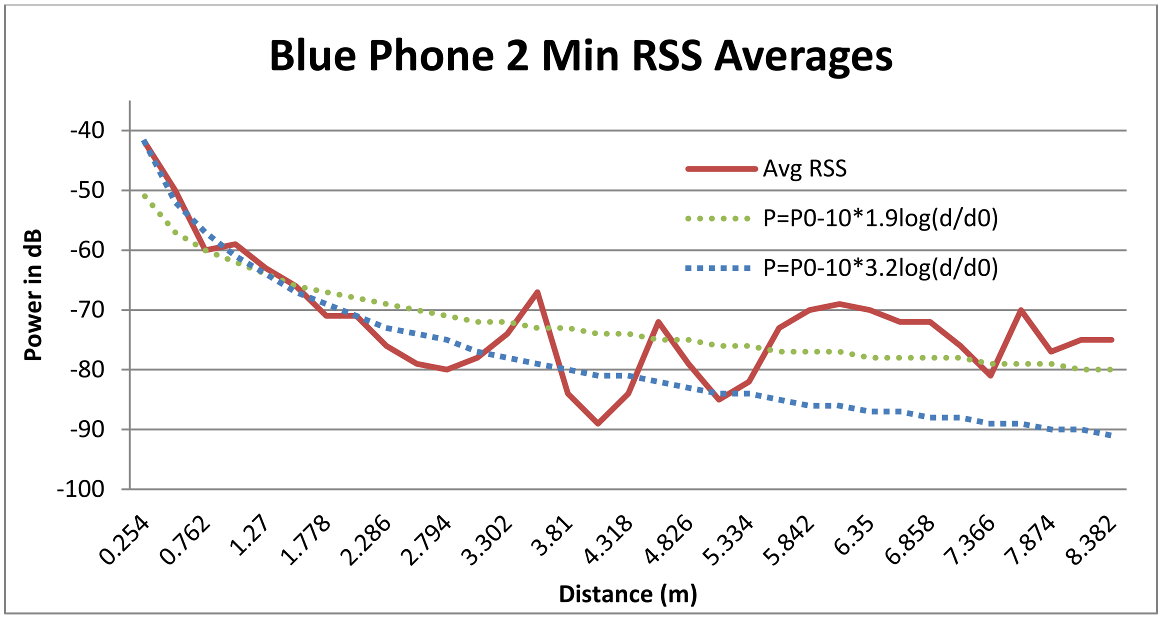


**Figure 11 - 2 min averages in hallway with unpredictable results after 4m**

Another recording experiment with the green phone is conducted a week later in the same location. Unfortunately, this data is highly unrealistic and completely unusable due to the high standard deviations at each distance. Figure 12 shows the graph of every RSS value recorded throughout the experiment. Few explanations can be offered for these results. There is minimal human traffic in the hallway and in general fewer people in the building than the previous experiment. The application does not record RSS values if it is not attached to the specific hotspot, so readings from another Wi-Fi source are unlikely. Regardless, the 2 minute averages are plotted and are shown in Figure 13. The green line shows the logarithmic trend line with a path-loss variable of 1 which is unrealistic. A slightly more probable trend line is shown in dotted red with a variable of 2.8.

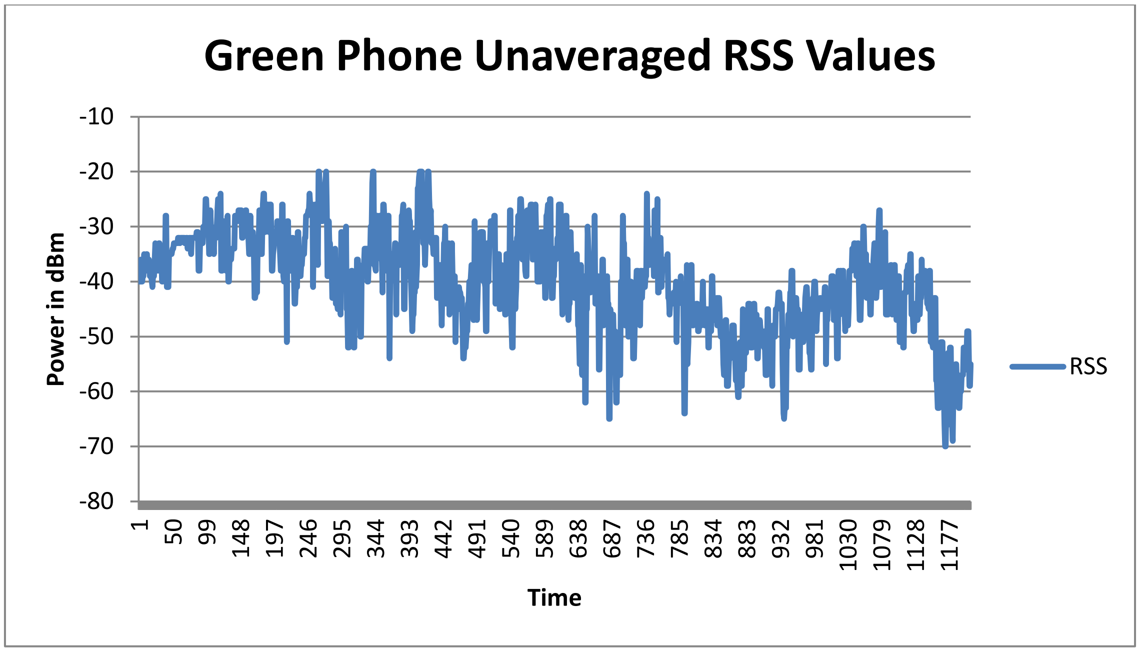


**Figure 12 - RSS recordings in hallway with another phone showing inaccurate readings**

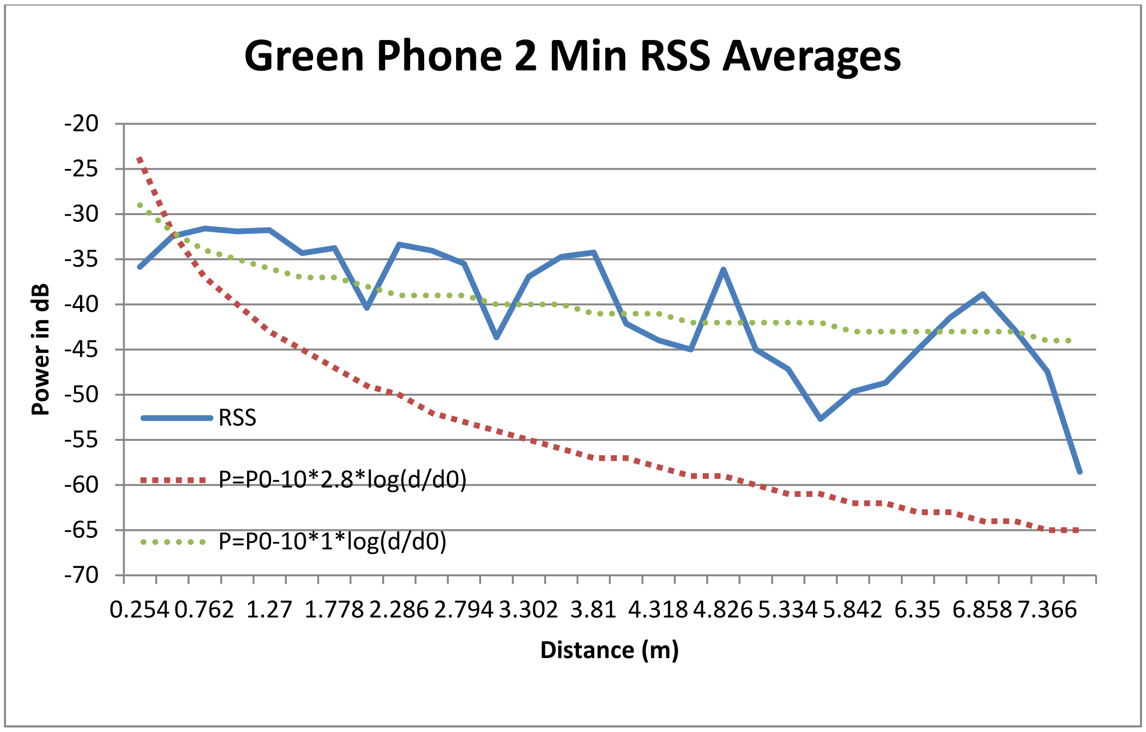


**Figure 13 - 2 min averages in hallway showing averaging does not reduce the inconsistencies of the results**

In an effort to assess the equipment in another location, another experiment is conducted in a laboratory space at AFIT. Both phones are tested under the same procedures as before. Figure 14 shows the green phone's RSS values while Figure 15 is the 2 minute average. Like the previous two figures, the RSS values are not typical for an indoor environment. While the trend line for Figure 15 is slightly more realistic, there is low correlation between the two graphs to make a generalization about signal strength to distance given different locations.

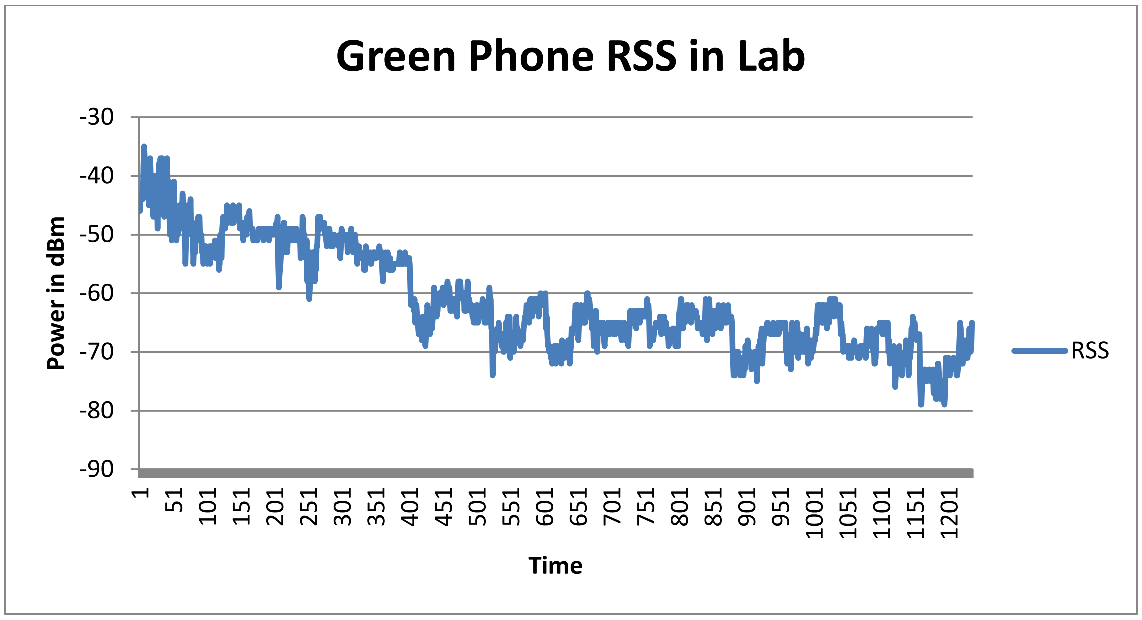


**Figure 14 - RSS recordings in a lab environment with a general logarithmic decay**

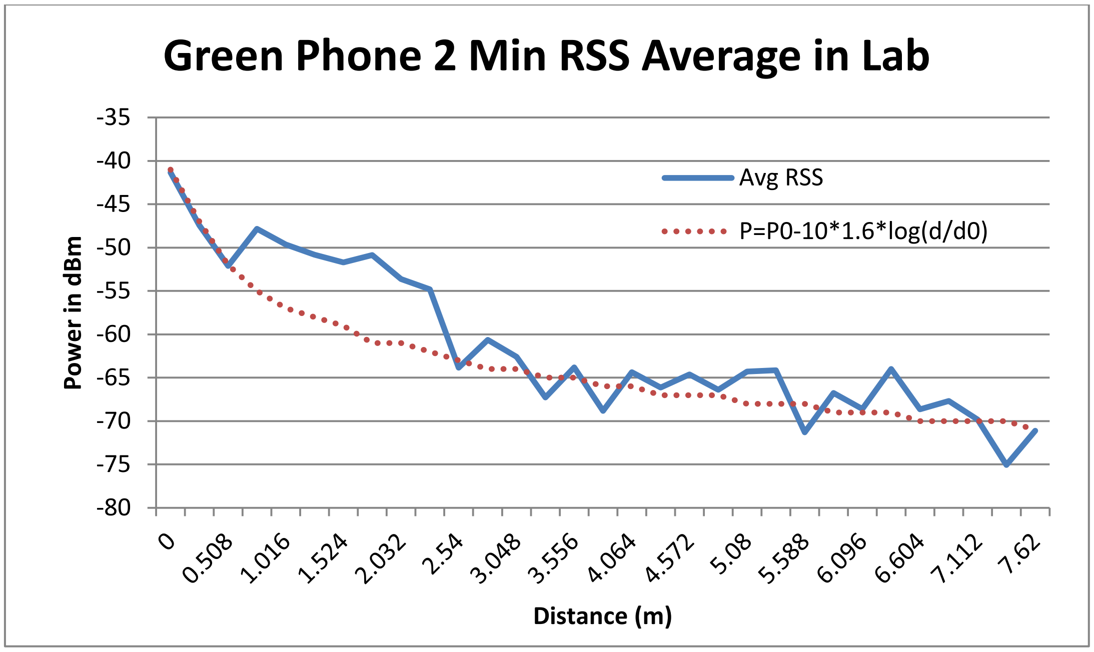


**Figure 15 – 2 min averages of the previous plot where the trend line does not suggest a realistic path loss exponent**

Testing the blue phone yields much better results. The RSS plot is shown in Figure 16. The correlation between distance and RSS value is much clearer than the other phone. The average plot, shown in Figure 17, also shows a much better trend line with a reasonable path-loss variable of 3.1. Viewing these plots along with Figure 10 and Figure 11 show a similar logarithmic decay followed by an increase received power. As a comparison, the data from Figure 11 is plotted with the data from Figure 17 to produce Figure 18. The RSS values for the hallway were lower than in the laboratory considering the latter contains many chairs and metal cubicles causing more multipath and signal loss through the space.

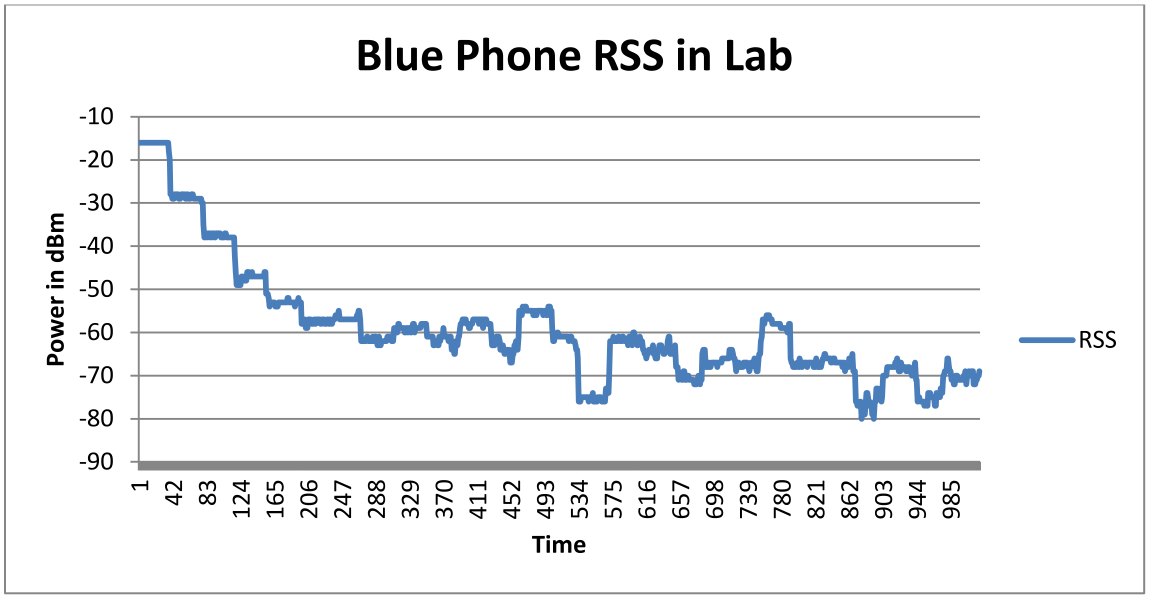


**Figure 16 - RSS recordings in a laboratory using a different phone**

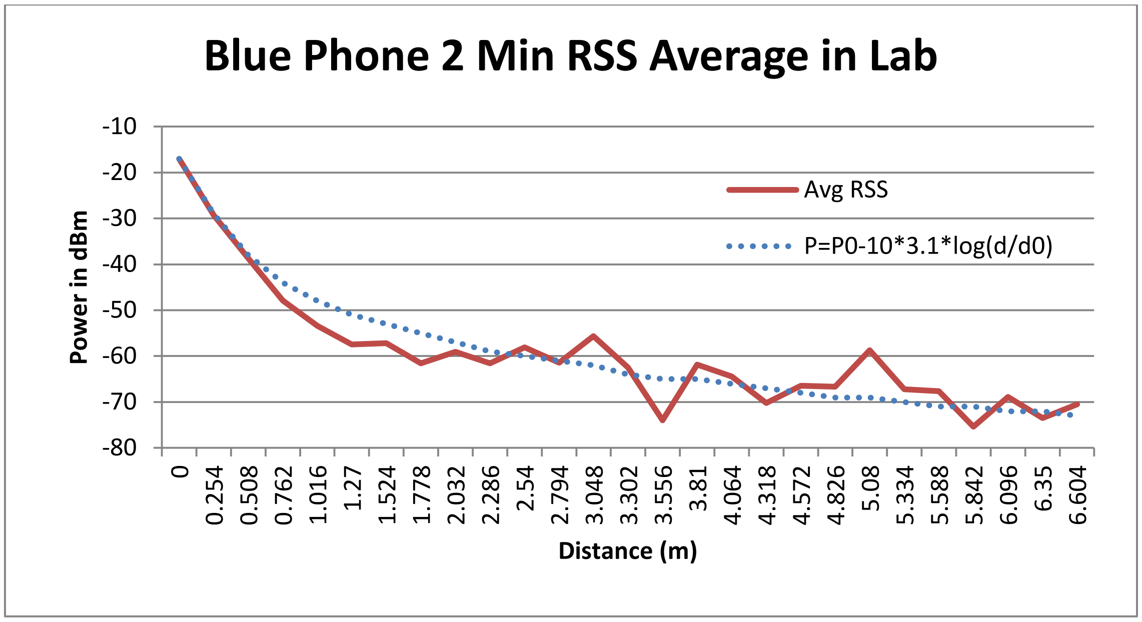


**Figure 17 - 2 min average of the previous plot with a much more realistic logarithmic decay**

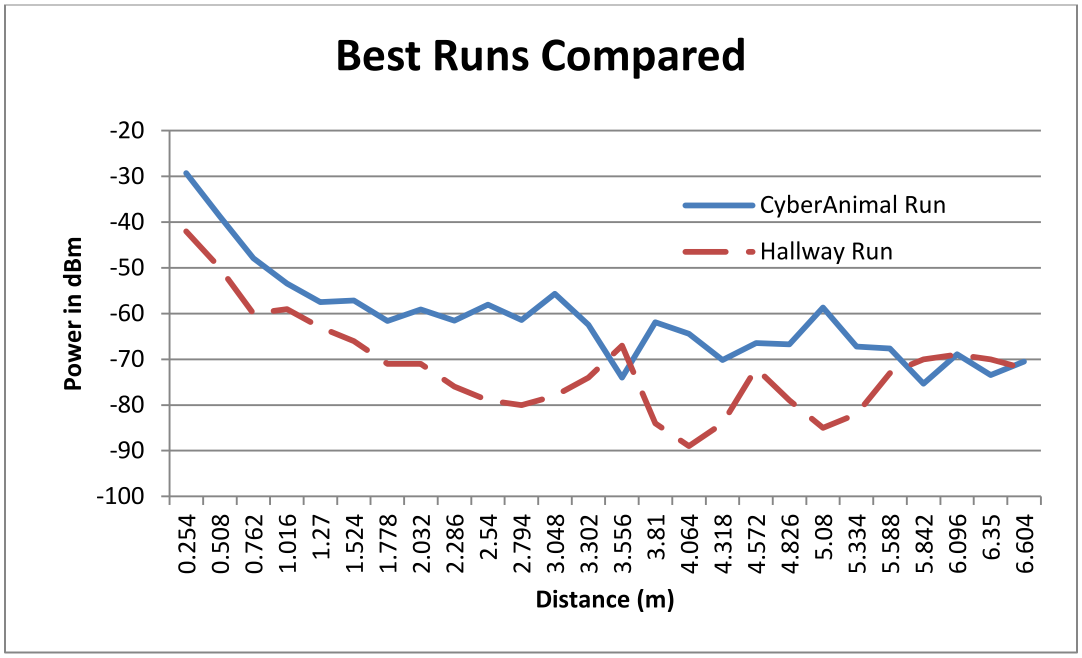


**Figure 18 - Two most similar runs compared showing a similar drop in RSS values at different distances**

Viewing these observations, it is clear that correlating distance to RSS measurements is potentially inaccurate. Other instances monitoring the RSS level while testing the application show that while not moving, the value varies several decibels and both HTC phones placed side by side could report an RSS reading 10dB apart.

In order to move forward with the experiment, another location is modeled, similar to the laboratory environment, with readings at the starting point of 0.254m then at every meter after. Since these readings are the locations that are going to be used for the experiment, recording at two minute intervals at only these measurements are more time effective. Recording the other distances in between the meter intervals only adds time necessary to complete the experiment and adds little clarity to the trend line needed to characterize the RF environment. Figure 19 shows the RSS values recorded followed by the next figure showing the average values along with the standard deviation for each location. Most variances are approximately 1dB, which is good for determining reliable RSS measurements for the specific location. The two worst measurements expectedly occur at the further distances resulting in variances of 3.7dB and 3.1dB for 6m and 7m respectively.

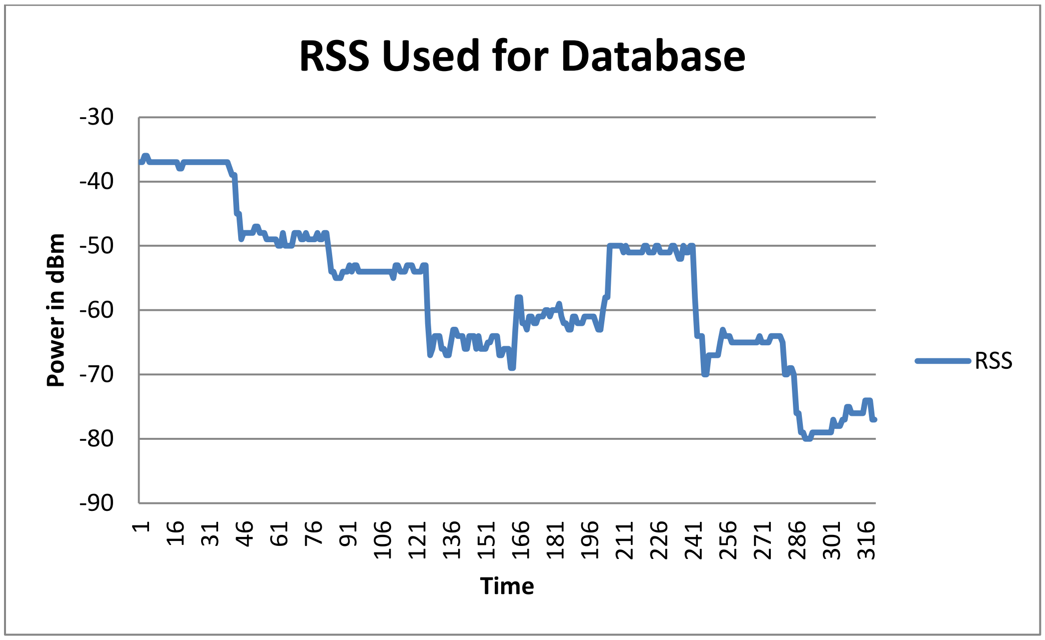


**Figure 19 - RSS recordings used for building the database**

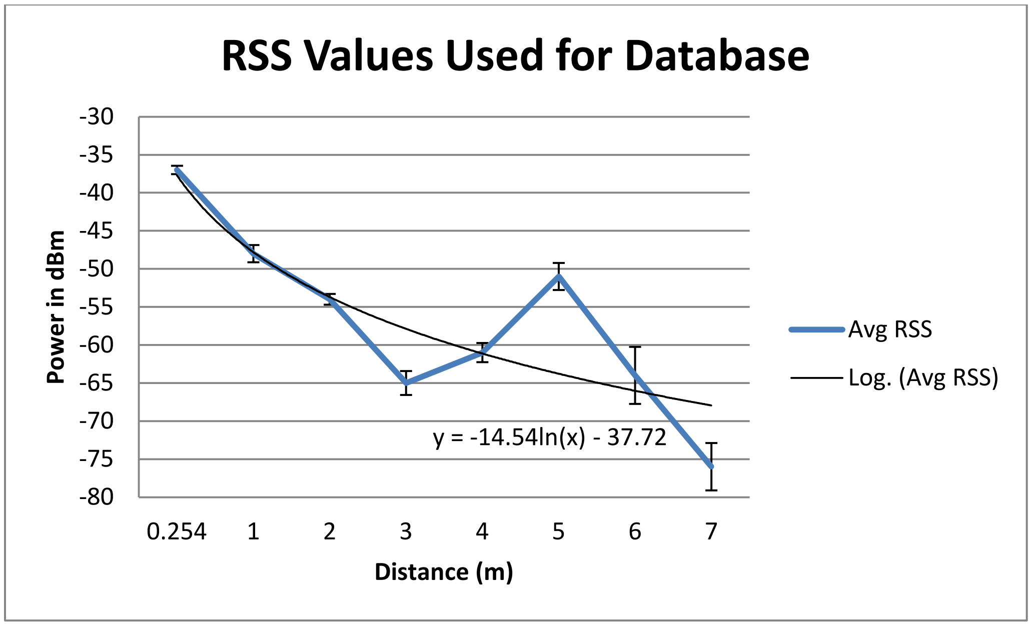


**Figure 20 - Averaged RSS values with standard deviations**

The RSS and distance values that are used to build the database are shown in Appendix B. These commands easily fill the database for lookup during the execution of the program. The first column represents the primary lookup key which is the RSS value, followed by the distance corresponding to that value and finally accuracy assigned to that reading. The database was constructed with a path-loss exponent of 2.2 in order to follow the logarithmic trend line as shown in Figure 20. The RSS range goes from -37 to -73 which capture the entire signal strength breadth for this experimental location.

As a comparison, three of the recordings from this section, all from the same blue phone, are plotted on the same graph with three potential path-loss curves. As covered in chapter 2, work found in (Rappaport, 2002) and (Tummala, 2005) describe the path-loss exponent $n_p$ to range from 1.26-1.8 for line-of-sight in buildings and 2.7-3.5 for urban areas. Using a starting power of -48dB at 1m, Equation 1 was used to generate the lines.

Because 802.11 transmitters tend to use different power levels dependent on the brand, there are no one-size-fits-all path-loss curves for WLAN signals.

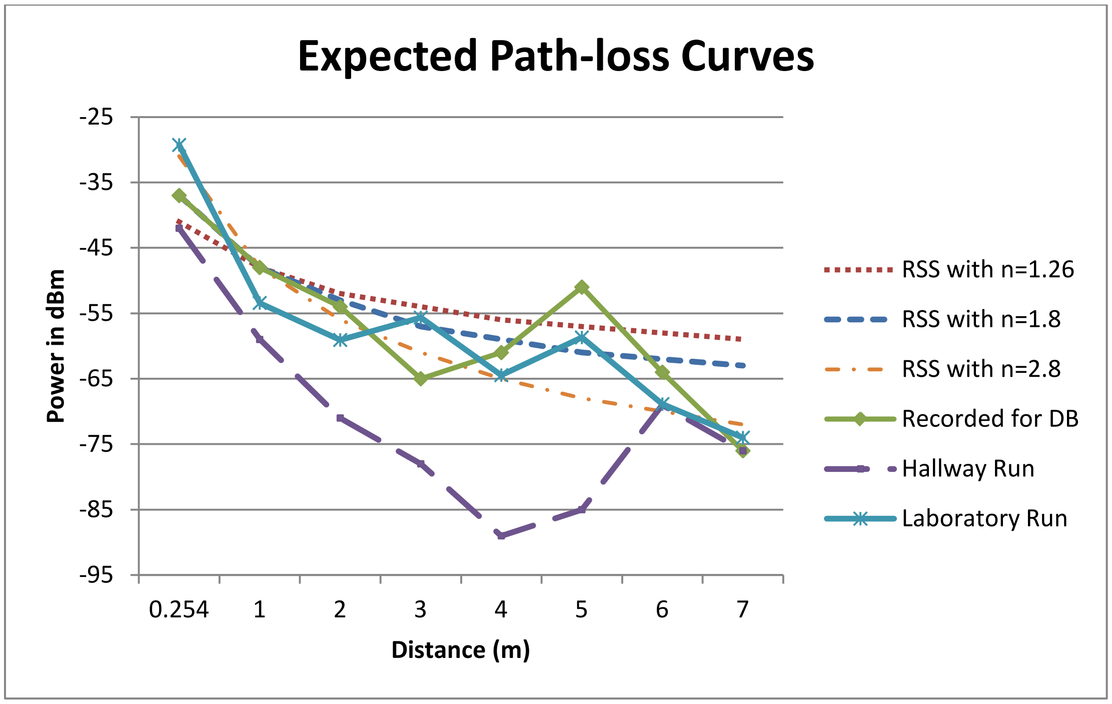


**Figure 21 - Expected path-loss curve estimates for 3 exponents compared to actual recordings**

### *4.2.2 Accelerometer*

As described in Chapters 2 and 3, the `SensorEventListener` produces a `SensorEvent` whenever the values change. In order to reduce the number of broadcasts regarding a change in a sensor value, a threshold of 1.5 is chosen in order to overcome slight, almost accidental, movements. For this research, X and Y axis readings tend to hover around -0.9 while remaining perfectly still on a table. The Z-axis is ignored as this effort does not intend to deal with three dimensions of localization. When the phone is lying flat and moved toward its right, the event values should be positive and negative for the other direction. This, however, is not always observed as seen in Figure 22.

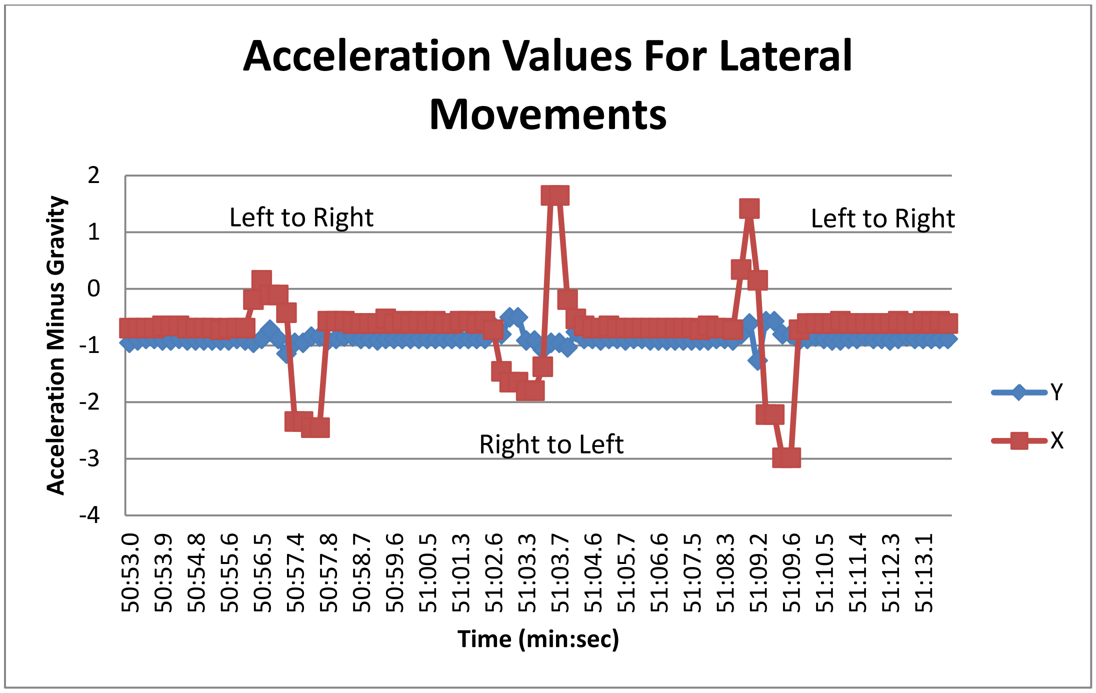


**Figure 22 - Acceleration X and Y values moving right to left to right**

While the readings are positive, at the end of the movement it is sometimes equaled with a negative value. This characteristic could be difficult to eliminate because the phenomenon does not always happen and could be a result of the phone legitimately moving in the opposite direction. Directly from the Android Developer `SensorEvent` webpage, “when the device lies flat on a table and is pushed on its left side towards the right, the *x* acceleration value is positive” (Google, 2010). The same situation is observed while moving the phone up and down the table, illustrated in Figure 23. Another quirk observed is when the phone moves in an arced path as shown in Figure 24. This motion is to test if the assumption that the phone will always face the hotspot could still produce worthwhile accelerometer data. With a significant movement, only a single

`SensorEvent` value is reported above the threshold. After these observations, the accelerometer has to be limited to just depicting when movement occurs and not necessarily how far or what direction.

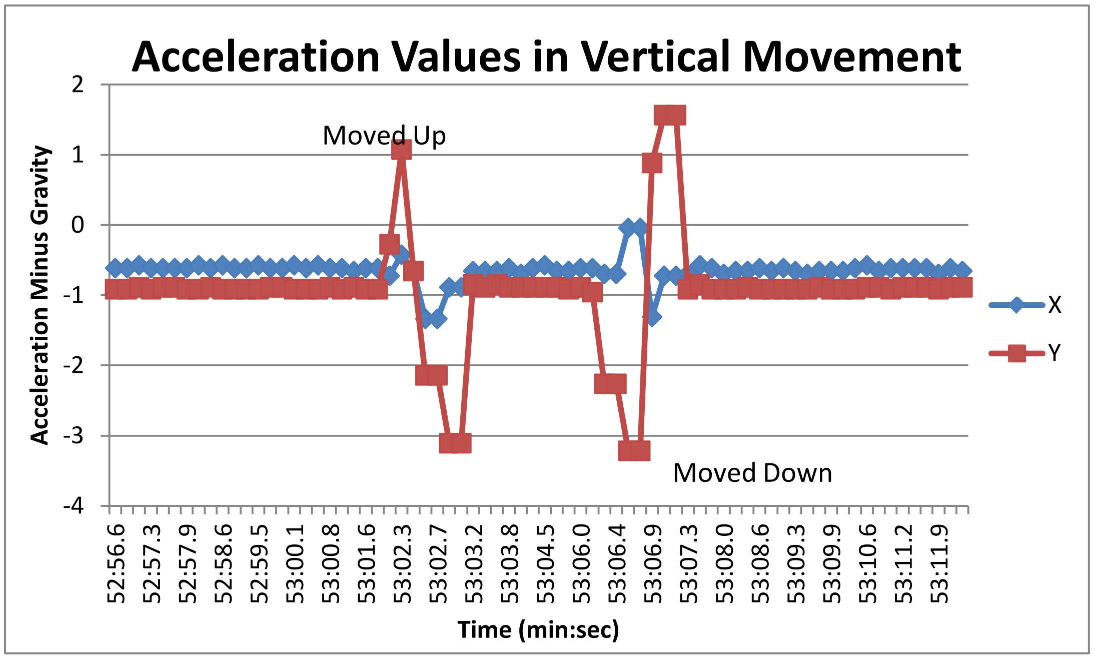


**Figure 23 - Acceleration X and Y values moving phone up and down**

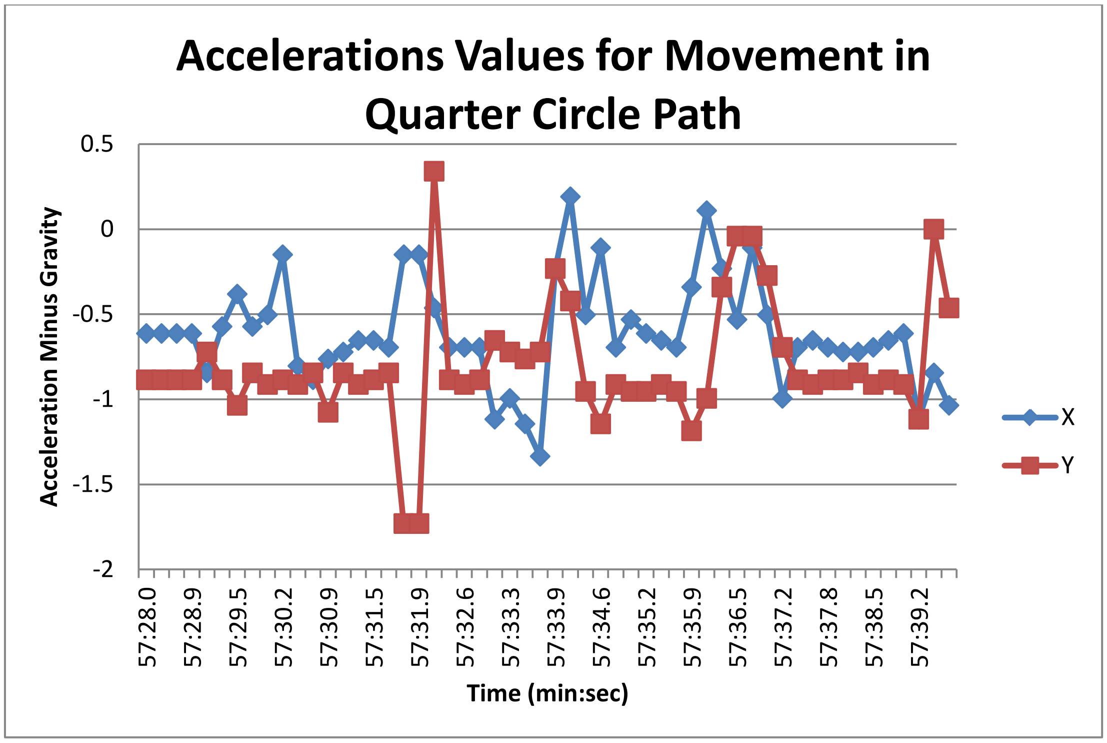


**Figure 24 - Acceleration X and Y values moving phone in quarter circle**

### *4.2.3 Compass*

The compass, used primarily for calculating locations given distances and other latitude and longitude points, exhibits similar inaccuracies as the accelerometer and WLAN adapter. While at rest on a flat table, the `SensorEvent` values can range +/- 5 degrees. This variance is not particularly limiting, however the internal sensor can be manipulated causing it to lose calibration. Other nearby metal objects can cause this interference without warning causing the system to produce erratic results. An example of this interference is shown in Figure 25. While the phone is lying on the table, another phone is moved toward the test phones' left side, over it by a margin of one inch, and exits the area to the right of the test phone. When the other phone is placed away from the test phone, it is observed that the stabilized compass value is now 41 degrees from the original observed azimuth. In order to mitigate this risk, the use of the compass is programmed into the system; however, a hardcoded value is used to calculate the actual distance.

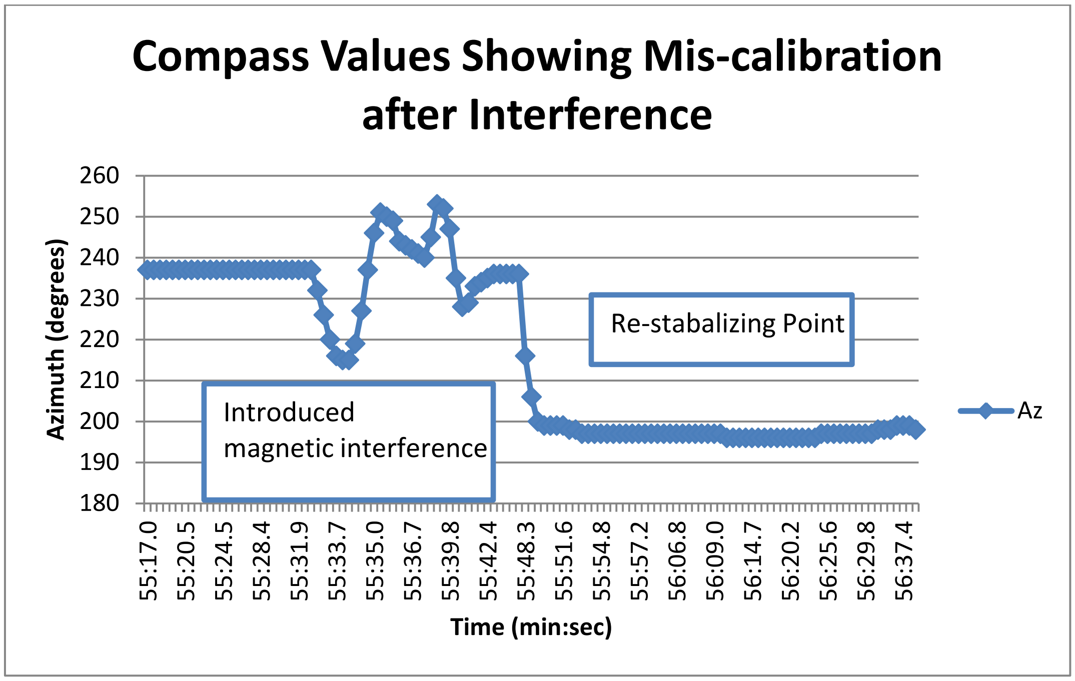


**Figure 25 - Compass values with interference added**

### *4.2.4 Overall Localization System*

As described in Chapter 3, eight locations are chosen for measurement. The phone is moved between two locations at a time and the location estimate is recorded. Figure 26 through Figure 33 show each of the locations recorded. The y-axis presents the individual experiment that is run while the x-axis represents the distance that is estimated. The vertical axis in each plot is positioned at the distance being measured to clearly depict what estimate is close to the testing distance, and which is further.

The 0.254m results first highlight how a difference in database training can become invalid. The database is trained with lower RSS readings at 0.254m but during experimentation, the RSS levels at that distance are higher, thus corresponding to a higher estimate, such as 0.652m. Figure 27 also shows where many estimates have a higher RSS reading thus corresponding to the estimates at 2m instead of 1m. The values at 1m have the third highest error of the experiments but the estimates are either clearly near 1m or 2m suggesting that the RSS readings cause the database lookups to go quickly from approximately 1m to 2m. Figure 28 shows the estimates tend to fall into four different columns. Like other plots with columnar results, the RSS values consistently grouped into different numbers. The lack of estimations in between the columns is due to having limited RSS values available to catalog every potential location. Appendix B contains the database used to achieve these results.

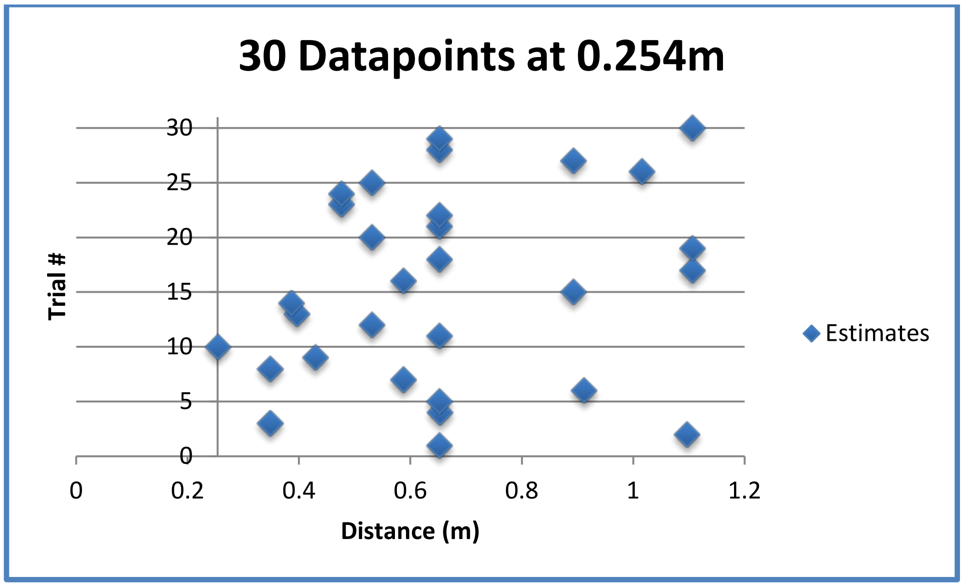


**Figure 26 - Estimations at 0.254m show the system over-estimates the position**

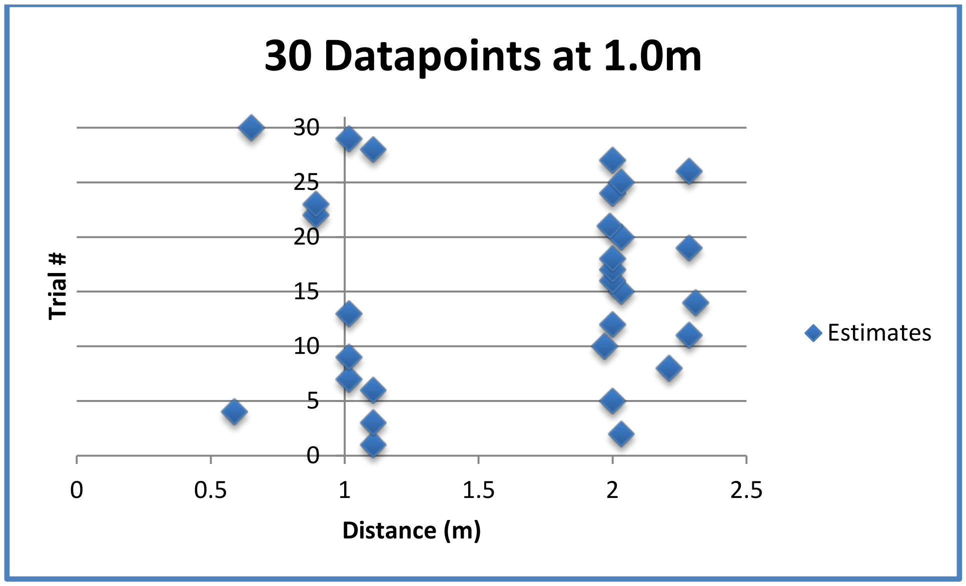


**Figure 27 - Estimations at 1m were split between near 1m or 2m**

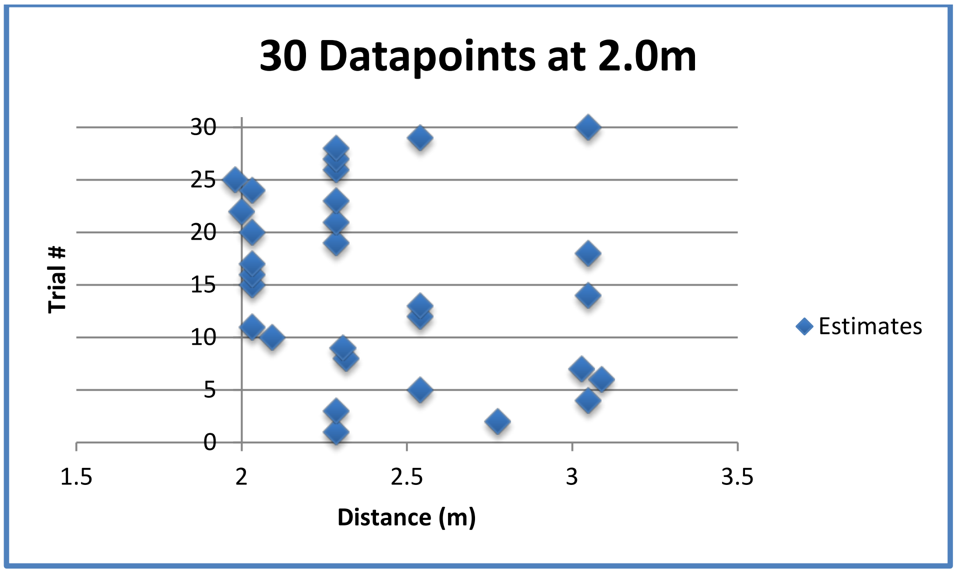


**Figure 28 - Estimations at 2m overestimate the distance**

The experiments run at 3m and 4m produce the closest average estimate of other distances, except 6m. Figure 29 and Figure 30 each show the distribution of their data. What is interesting is that the 3m run also produces the highest standard deviation meaning the estimate is the most unreliable of any other distance estimate.

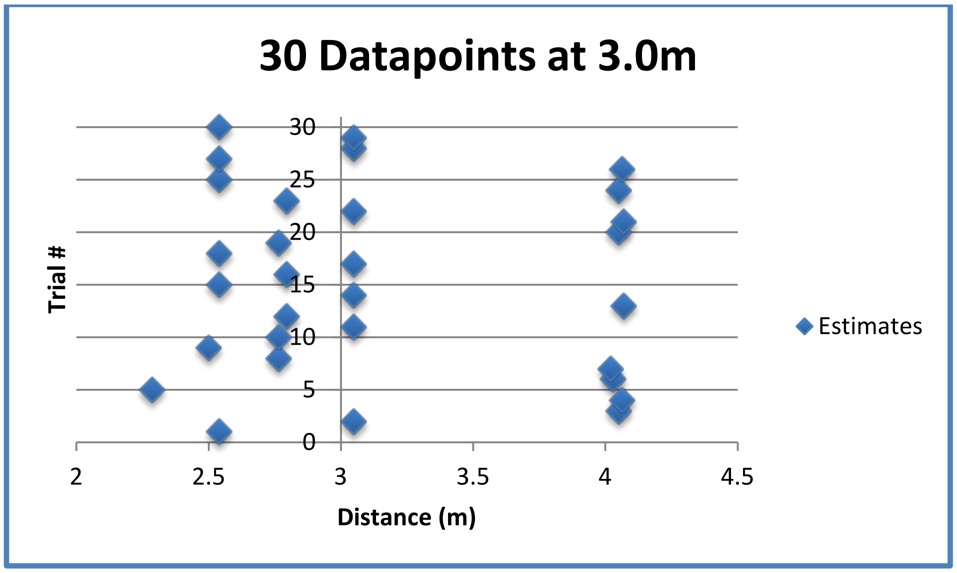


**Figure 29 - Estimations at 3m are fairly close to the actual distance**

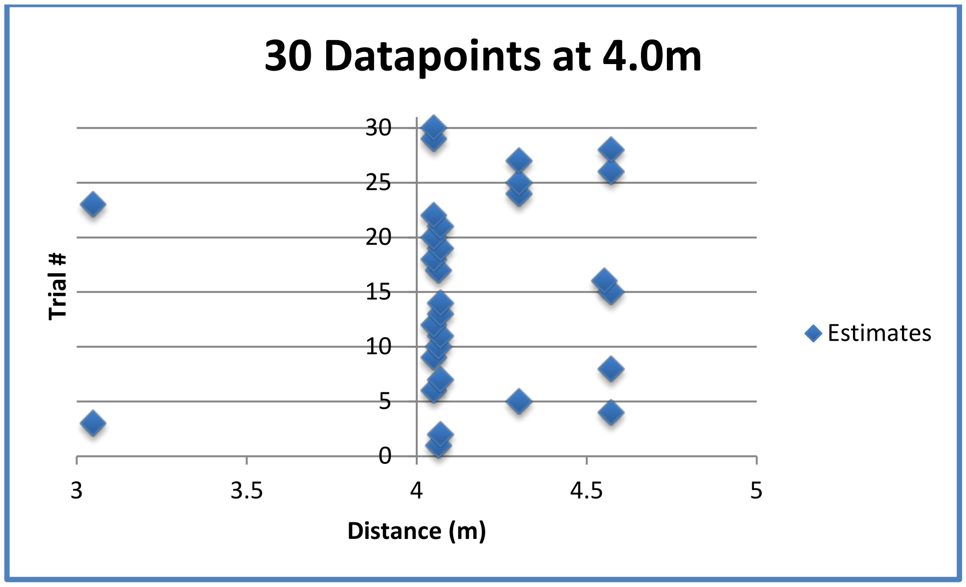


**Figure 30 - Estimations at 4m were close to the actual distance**

Figure 31 shows the largest average error found. This is mostly related to the difficulties in accounting for the RSS reading between 4-5m as shown previously in Figure 18. The best thing to overcome this problem would be to program conditions that would be illogical such as if the previous estimate was at 4m and significant movement was detected, then the next estimate should be 5m for that RSS value. Figure 32 shows that results for 6m are the closest to the actual distance; however, a few early trials reported estimates of 4.3m. The low estimate is a result of the phone calculating higher signal strength than what it should have, possibly caused by not refreshing adequately for the new distance. If these three trials are removed or rerun, the 6m results could potentially go from the second highest standard deviation to the lowest making it the best distance estimate. In the worst case, the three reruns would produce the same estimate of 4.3m and in the best case; an estimate much closer to 6m could be produced. The final distance, 7m, represents almost a give and take aspect of this whole experiment. All estimates are shorter than the actual distance showing that the RSS value was not as weak as anticipated at this distance. However, if the database is adjusted to represent -67dB as 7m instead of 6.2m, then the average estimate and standard deviation for the 6m distance would likely go up drastically making it more unreliable.

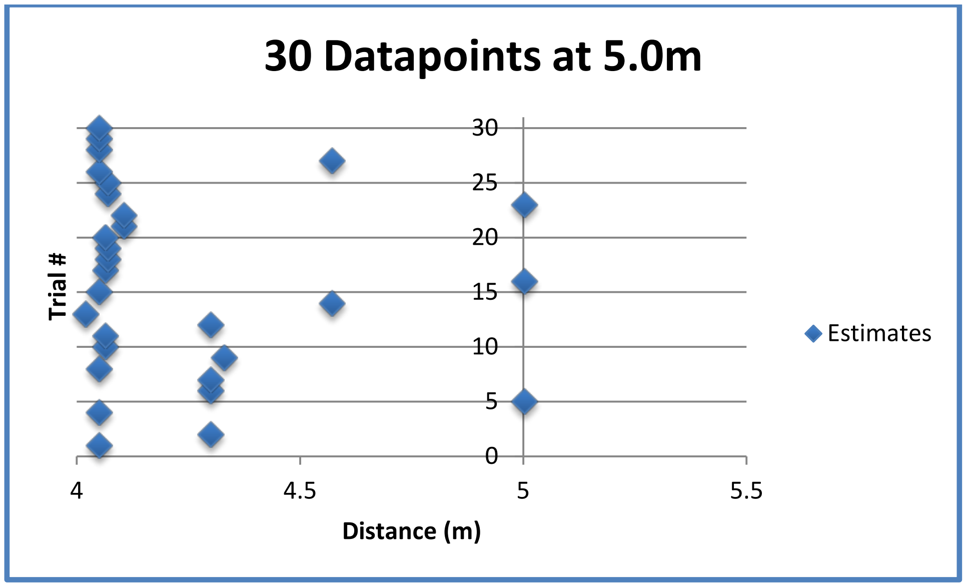


**Figure 31 - Values at 5m are underestimated**

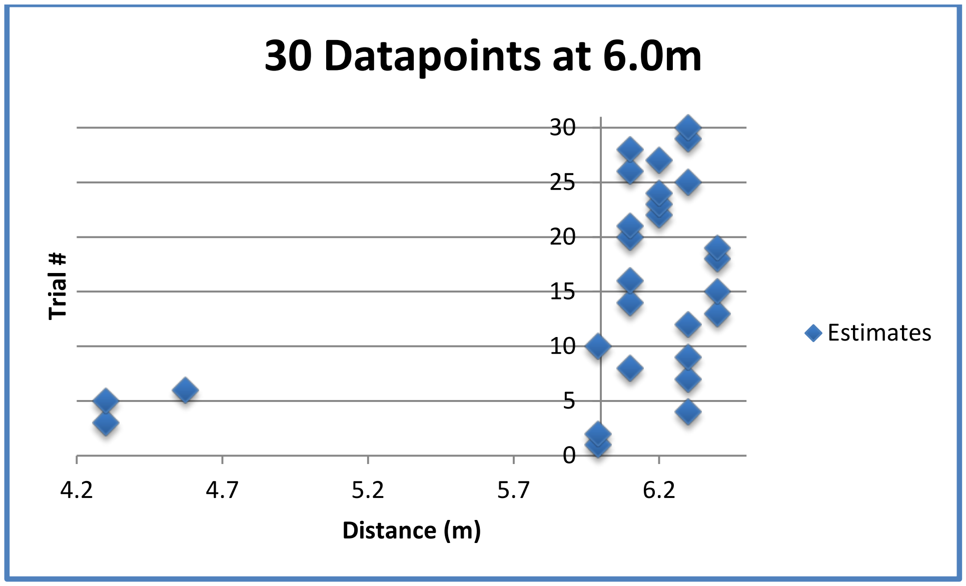


**Figure 32 - Estimations at 6m were closest to the actual distance**

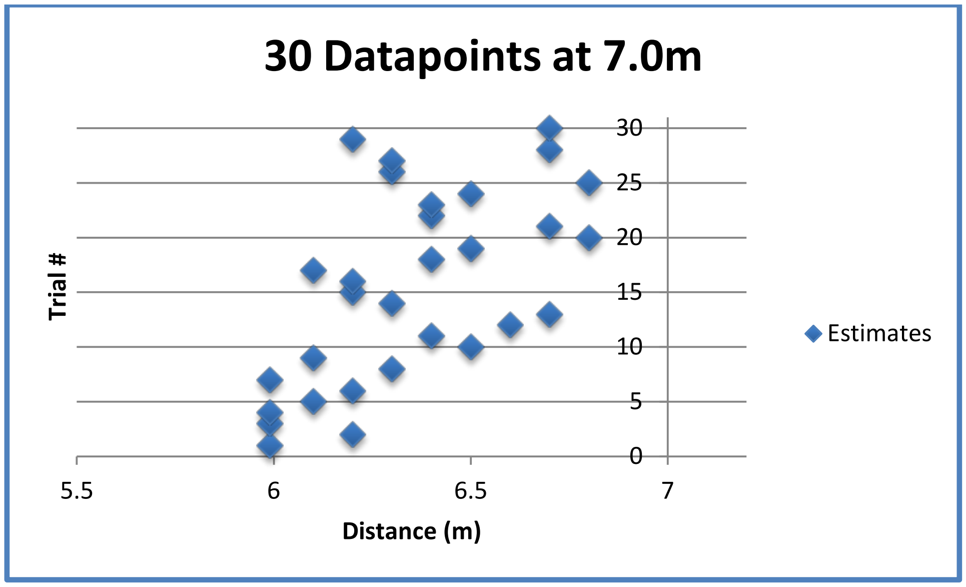


**Figure 33 - Estimations at 7m were all underestimates**

In order to summarize the results, the average error rate for each distance interval is plotted with standard deviation error bars. This plot is shown in Figure 34. The horizontal axis indicates what average estimates are closer or further from the access point. For instance, the average estimate for 0.254m is 0.593m meaning the average estimate is on the "high" side. By contrast, most of the results for the 7m distance are around 6.1m so the average estimate is on the "low" side and closer to the access point rather than further from it.

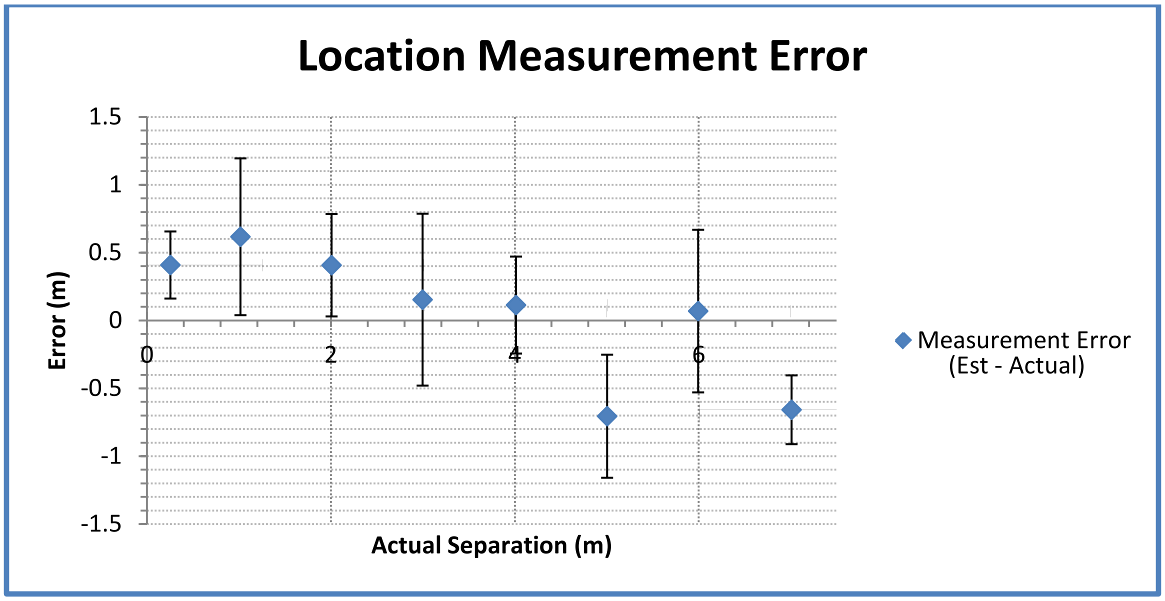


**Figure 34 - Error plot for each distance value showing if the average estimate was an over or under estimate and how widespread the standard deviation was**

Overall, the system tends to overestimate the distance between node and access point at closer distances and underestimate at longer distances. As seen from the individual graphs and the standard deviation, the larger variances occur around 1m, 3m, and 6m with values of 0.578m, 0.633m, and 0.599m respectively. This effect is due to the signal fluctuations explored in section 4.2.1 where the RSS value oddly gets stronger as distance increases, then falls sharply only to rise again. This feature makes it very difficult to distinguish between these distances accurately. A potential reason for the increase in RSS is that as the wireless hotspot receives requests from the test phone it increases its output power, but is unable to keep that output level.

### *4.2.5 Worst Case Scenarios for Distance Estimation*

With any localizing method there will be unavoidable errors. In order to better understand how badly the estimate could be with this system, some of the worst estimates are plotted with error attributed to GPS readings. The worst values at 0.254m, 2m, 4m, and 6m are chosen coupled with the worst GPS error for that distance. Figure 35 shows a plot of each location with the original estimate from the previous section, where the GPS could potentially be, and where the actual location is. In instances where the worst estimate is closer to the anchor (estimates at 4m and 6m), the potential anchor position is positive 1.8m. In the other scenarios, the anchor position is negative 1.8m to make the overall separation more drastic. The location with the lowest potential error is at 0.254m with an error of 2.65m and the largest potential error is at 6m with an error of 3.5m.

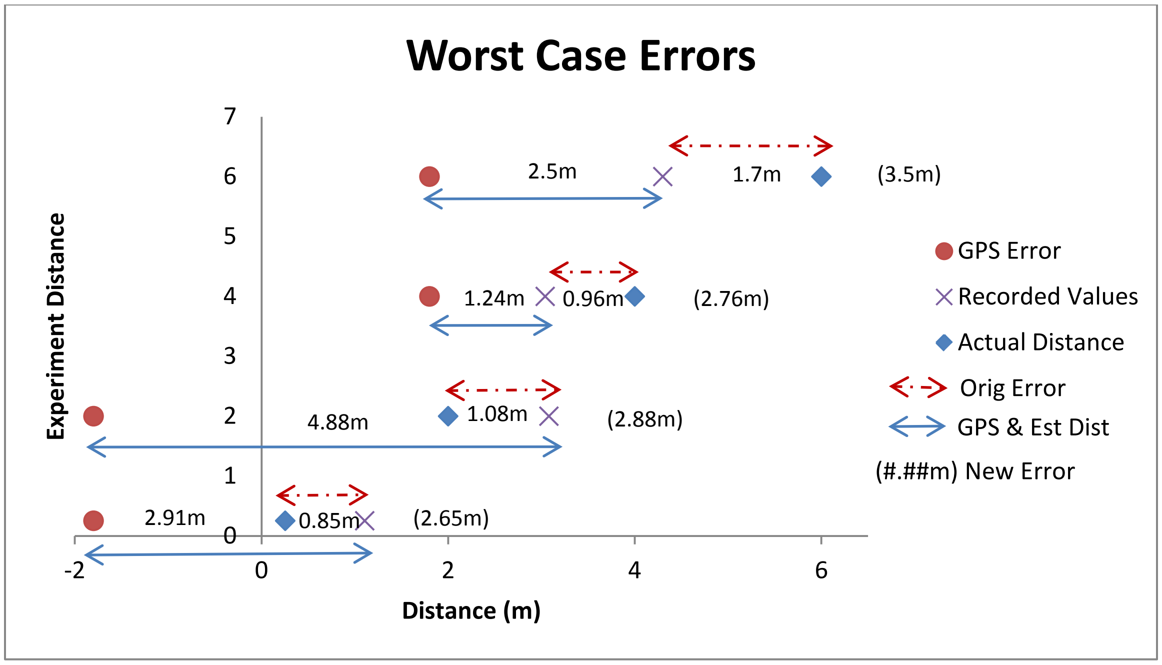


**Figure 35 - Worst Case Estimations for 0.254m, 2m, 4m, and 6m**

These errors consider keeping the recorded estimate the same, but since distance is a function of the RSS value, moving the anchor in either direction would most likely affect the RSS that the node gets. However, if the RSS database is constructed based off a false anchor position, the resulting estimate of a non-localized node is better than no estimate at all. For instance, if a node has a stale position because it went from outside to 10 meters inside, it would join the localizing network and potentially think it is 2.5m from the anchor as suggested in Figure 35. This results in an error of 3.5m, whereas if the node kept its previous location, the error would be 4m. More cooperating nodes could reduce the error further.

## 4.3 Investigative Questions Answered

In order to determine if commercial mobile phones can be used for cooperative localization without relying on an external infrastructure, the capabilities and characteristics of the phone need to be understood and modeled. By testing individual sensors, their virtues and shortcomings can be better implemented into the system. In a restricted experiment, results found that cellular phones can cooperatively localize one another when one node has a bad location estimate while another has a much better one.

One of the goals of the research is to perform better than previous efforts. Comparing the results of this effort, which relied on a much smaller scale than the systems presented in Chapter 2, to the other systems is perhaps unpredictable. In order to accurately compare them, either the other systems would have to be reduced to one access point, or multiple access points would have to be added to this work. However, looking at the preliminary results, this system fared well with errors less than a meter. The other efforts which used multiple sensors are CERP, DMRF, and SELFLOC (Gwon, Jain, & Kawahara, 2004) (Fang & Lin, 2010). All of them have accuracy ranging between 1-2m and error rates less than 2m. This effort found accuracy less than a meter and error rates below 0.65m. The system listed in Chapter 2 to do better was Ubisense which is a commercial product using special tags that operate using UWB signal. The stark difference in accuracy is directly related to the higher precision available to UWB signal and special hardware to compute both AOA and TDOA versus WLAN which, in this effort, only relies on RSS.

## 4.4 Summary

This chapter presents the data collected from the various sensors sought to aid in cooperative localization. By analyzing the merits of each sensor available, the confidence of a resulting location can be achieved. The results of characterizing the RSS environment have shown that variations in measurements are too drastic to accurately correlate a distance to the measurement. When considering acceleration data, only magnitude has been shown to be valuable for localization. While the compass showed a few degrees of error, the primary concern was keeping the sensor calibrated.

# V. Conclusions and Recommendations

## 5.1 Chapter Overview

This chapter concludes the research presented in this thesis. Section 5.2 discusses the overall conclusions of the research and what has been achieved from the original goals. Section 5.3 discusses the importance of this effort while Section 5.4 explores some suggestions for future research. Finally, Section 5.5 summarizes the chapter.

## 5.2 Conclusions of This Research

This research seeks to bring cooperative localization to computationally constrained devices placing the entire system onto the mobile node in real time as opposed to relying on post processing and extra hardware. Location estimates are produced within three seconds of moving to a new location instead of having to wait until after the experiment is complete and the data is processed. Several key observations are achieved:

- Sources of RSS variances include time of day and specific device being used
- RSS values tend to float a couple values even when neither device moves
- The compass can easily lose calibration
- The accelerometer shows multiple directions of movement even while controlled to one direction

As theorized, the accelerometer has to function as a method to detect when movement occurs, not necessarily what direction or how far. This short coming is logical since the accelerometers chosen for smart phones, particularly lower end models, are never intended for high precision activities. The capabilities of the compass and RSS are more lacking than originally expected. Calibration routines would need to be worked into the testing methodology before every experiment run to ensure the hardware is performing as expected. RSS limitations could be overcome with multiple sources contributing readings to reduce error and fluctuations. Also, the lack of directly being able to read the signal strength level on the Bluetooth signal prevents it from being readily used for cooperative localization. The system could be adapted by modifying the kernel to make this information available. Another approach could be if a phone has the ability to connect to another via Bluetooth, then the distance must be within a certain range.

## 5.3 Significance of Research

The capabilities of GPS and cellular tower triangulation are widely known when it comes to location determination. The research done in the realm of cooperative localization is also well known with robotics and specialized hardware. This research examines the potential of localizing nodes without special hardware and entirely amongst themselves. By exploring the various sensors available on common smart phones, the merits of using them could be weighed appropriately. For instance, the variation in wireless signal can be averaged if the phone has not been moved according to the accelerometer. The compass can be used to compute the latitude and longitude of another node it is pointing towards. When incorporating the JDL Data Fusion model, sensors can be used as they are available or as they can contribute to the improvement of the estimation.

Every effort to minimize location error can aid in combat operations, search and rescue missions, and even advertising. This location minimization becomes more possible since more and more people are adopting smart phones as their primary communication device. The multitasking nature of Android allows for the localizing services to run seamlessly in the background while the user is able to operate their phone as they require. The Android operating system allows them to be a passive node in the network helping other nodes to localize, or switch to being an active node seeking directions or distance information to other nodes.

## 5.4 Recommendations for Future Research

The variance of RSS in different locations and between nodes prevent the use of this signal from reaching its desired functionality. Future work could do more to characterize the ability of a phone to work as a Wi-Fi hotspot. There are undoubtedly differences between a standard wireless access point and a smart phone and understanding how they differ could result in a more robust database. Putting more modularity into the database would also be a worthwhile endeavor. As conditions change from one location to another, the path-loss equation should also change. This could reduce the error in distance measurements.

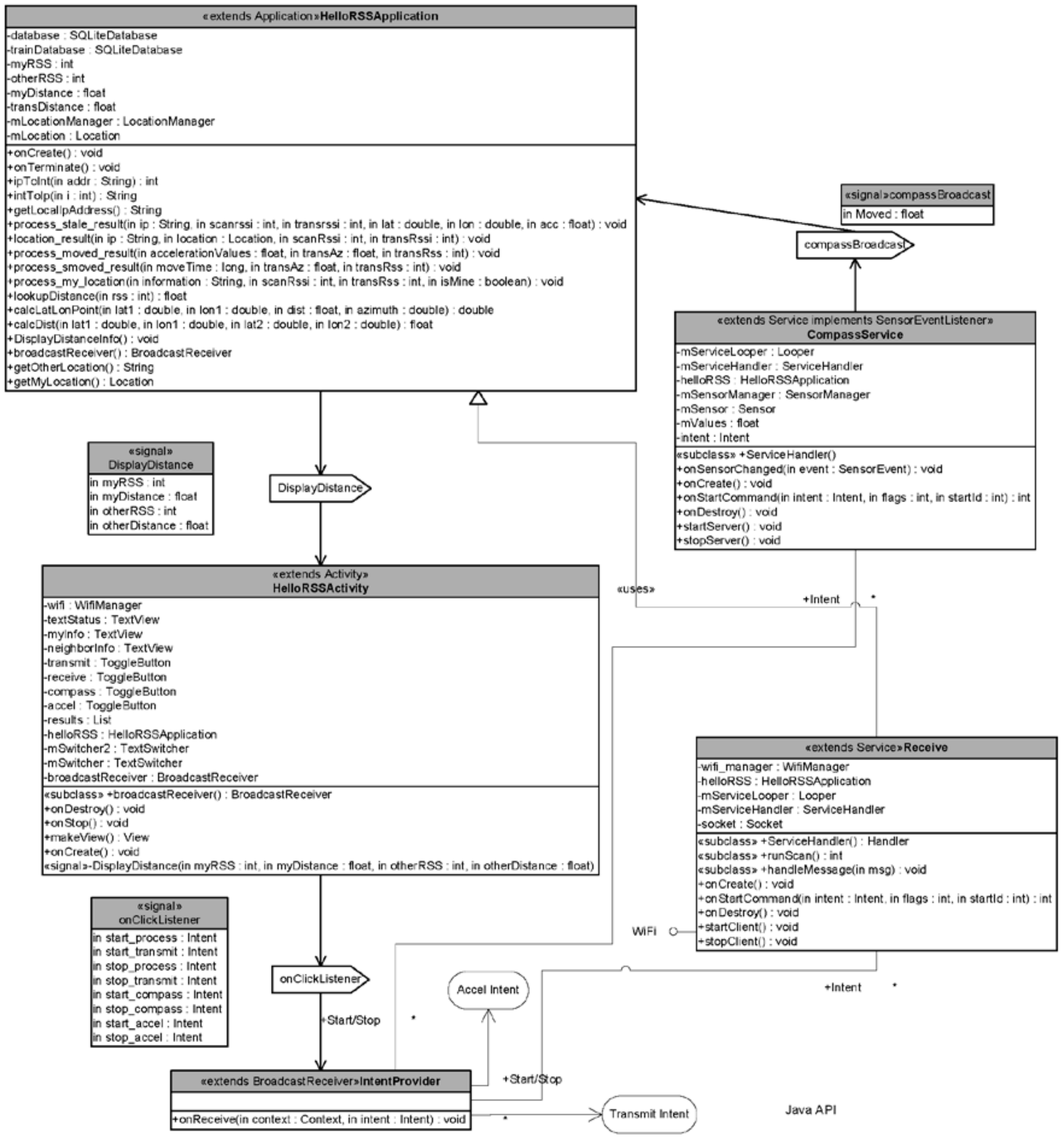
This research deals with a very sparse network of nodes which makes localization much more difficult. By adding more nodes, error can be reduced. Future research should include modifying the current framework to account for more nodes. Several assumptions are made to localize only two nodes. One of the considerations that would need to be addressed is how to add and access values in the database. The database requires unique keys for entries which means the IP address can not necessarily be used if you want to keep a catalog of the various locations a node has been. The RSS value also cannot be used because there could be numerous nodes at numerous locations with the same value. The database is conceived as a means to quickly localize nodes and can be kept for that reason, but it would just need a few adjustments such as creating a new unique key based off the RSS value and IP address of the neighbor node.

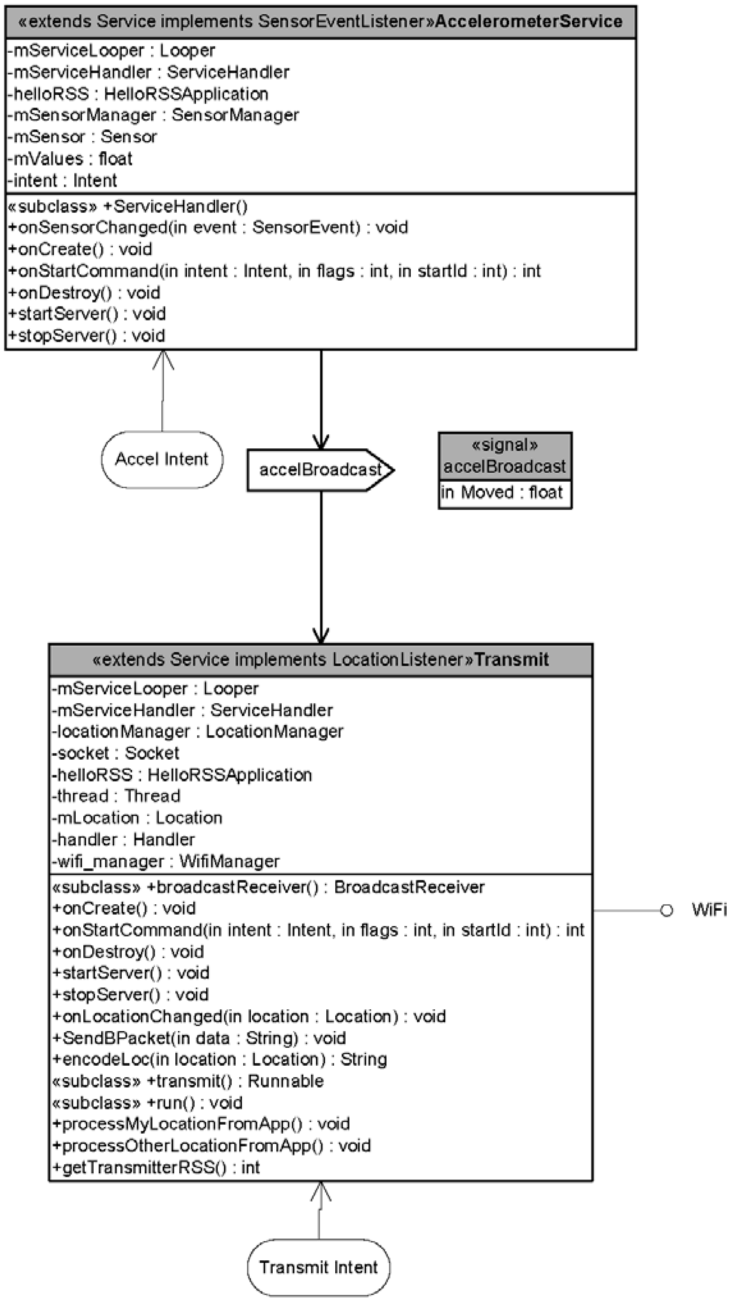
Ad hoc networks can expand a typical wireless footprint as large as there are nodes. A couple protocols (AODV and OLSR) have been attempted on Android and in the latest operating system release, 4.0, native support for peer-to-peer connections are implemented. By implementing an ad hoc network, nodes can route messages among themselves and not require a specific, stationary Wi-Fi access point. As noticed from testing the capability of a phone to operate as a hotspot, the transmission distance is not nearly as far as other infrastructure-based access points. The multi-hop packet routing would allow for a node potentially around the corner to become aware of the other nodes in the network. This network extension would be more beneficial for some of the scenarios described in Chapter 1 where the existing infrastructure may not be available and yet nodes need to be localized for critical safety reasons.

## 5.5 Summary

This chapter expounds the concluding remarks for this research effort. Overall, the ambitious tasks were difficult to achieve due to unexpected hardware limitations. More time had to be devoted to learning and understanding the programming environment and what the hardware was capable of instead of refining a localization algorithm. Carrying out physical experiments were also time consuming as pieces of software were tested, new issues regarding the environment were uncovered. Overall several key lessons were learned and a beginning towards establishing a cooperative localization system on a mobile platform is underway.

# Appendix A


«extends Application»HelloRSSApplication
-database : SQLiteDatabase
-trainDatabase : SQLiteDatabase
-myRSS : int
-otherRSS : int
-myDistance : float
-transDistance : float
-mLocationManager : LocationManager
-mLocation : Location
+onCreate() : void
+onTerminate() : void
+ipToInt(in addr : String) : int
+intToIp(in i : int) : String
+getLocalIpAddress() : String
+process_stale_result(in ip : String, in scanrssi : int, in transrssi : int, in lat : double, in lon : double, in acc : float) : void
+location_result(in ip : String, in location : Location, in scanRssi : int, in transRssi : int) : void
+process_moved_result(in accelerationValues : float, in transAz : float, in transRss : int) : void
+process_smoved_result(in moveTime : long, in transAz : float, in transRss : int) : void
+process_my_location(in information : String, in scanRssi : int, in transRss : int, in isMine : boolean) : void
+lookupDistance(in rss : int) : float
+calcLatLonPoint(in lat1 : double, in lon1 : double, in dist : float, in azimuth : double) : double
+calcDist(in lat1 : double, in lon1 : double, in lat2 : double, in lon2 : double) : float
+DisplayDistanceInfo() : void
+broadcastReceiver() : BroadcastReceiver
+getOtherLocation() : String
+getMyLocation() : Location
«signal»compassBroadcast
in Moved : float
compassBroadcast
«extends Service implements SensorEventListener»
CompassService
-mServiceLooper : Looper
-mServiceHandler : ServiceHandler
-helloRSS : HelloRSSApplication
-mSensorManager : SensorManager
-mSensor : Sensor
-mValues : float
-intent : Intent
«subclass» +ServiceHandler()
+onSensorChanged(in event : SensorEvent) : void
+onCreate() : void
+onStartCommand(in intent : Intent, in flags : int, in startId : int) : int
+onDestroy() : void
+startServer() : void
+stopServer() : void
«signal»
DisplayDistance
in myRSS : int
in myDistance : float
in otherRSS : int
in otherDistance : float
DisplayDistance
«uses»
+Intent
*
«extends Activity»
HelloRSSActivity
-wifi : WifiManager
-textStatus : TextView
-myInfo : TextView
-neighborInfo : TextView
-transmit : ToggleButton
-receive : ToggleButton
-compass : ToggleButton
-accel : ToggleButton
-results : List
-helloRSS : HelloRSSApplication
-mSwitcher2 : TextSwitcher
-mSwitcher : TextSwitcher
-broadcastReceiver : BroadcastReceiver
«subclass» +broadcastReceiver() : BroadcastReceiver
+onDestroy() : void
+onStop() : void
+makeView() : View
+onCreate() : void
«signal»-DisplayDistance(in myRSS : int, in myDistance : float, in otherRSS : int, in otherDistance : float)
«extends Service»Receive
-wifi_manager : WifiManager
-helloRSS : HelloRSSApplication
-mServiceLooper : Looper
-mServiceHandler : ServiceHandler
-socket : Socket
«subclass» +ServiceHandler() : Handler
«subclass» +runScan() : int
«subclass» +handleMessage(in msg) : void
+onCreate() : void
+onStartCommand(in intent : Intent, in flags : int, in startId : int) : int
+onDestroy() : void
+startClient() : void
+stopClient() : void
WiFi
«signal»
onClickListener
in start_process : Intent
in start_transmit : Intent
in stop_process : Intent
in stop_transmit : Intent
in start_compass : Intent
in stop_compass : Intent
in start_accel : Intent
in stop_accel : Intent
onClickListener
+Start/Stop
*
Accel Intent
+Intent
*
«extends BroadcastReceiver»IntentProvider
+onReceive(in context : Context, in intent : Intent) : void
+Start/Stop
*
Transmit Intent
Java API


«extends Service implements SensorEventListener»**AccelerometerService**
-mServiceLooper : Looper
-mServiceHandler : ServiceHandler
-helloRSS : HelloRSSApplication
-mSensorManager : SensorManager
-mSensor : Sensor
-mValues : float
-intent : Intent
«subclass» +ServiceHandler()
+onSensorChanged(in event : SensorEvent) : void
+onCreate() : void
+onStartCommand(in intent : Intent, in flags : int, in startId : int) : int
+onDestroy() : void
+startServer() : void
+stopServer() : void
Accel Intent
accelBroadcast
«signal»
accelBroadcast
in Moved : float
«extends Service implements LocationListener»**Transmit**
-mServiceLooper : Looper
-mServiceHandler : ServiceHandler
-locationManager : LocationManager
-socket : Socket
-helloRSS : HelloRSSApplication
-thread : Thread
-mLocation : Location
-handler : Handler
-wifi_manager : WifiManager
«subclass» +broadcastReceiver() : BroadcastReceiver
+onCreate() : void
+onStartCommand(in intent : Intent, in flags : int, in startId : int) : int
+onDestroy() : void
+startServer() : void
+stopServer() : void
+onLocationChanged(in location : Location) : void
+SendBPacket(in data : String) : void
+encodeLoc(in location : Location) : String
«subclass» +transmit() : Runnable
«subclass» +run() : void
+processMyLocationFromApp() : void
+processOtherLocationFromApp() : void
+getTransmitterRSS() : int
WiFi
Transmit Intent

# Appendix B

```
CREATE TABLE rss (rss integer primary key, dist float, accuracy float);
INSERT INTO "rss" VALUES(-73,7.010,1.150);
INSERT INTO "rss" VALUES(-72,7.010,0.416);
INSERT INTO "rss" VALUES(-71,7.010,0.416);
INSERT INTO "rss" VALUES(-70,7.000,0.635);
INSERT INTO "rss" VALUES(-69,6.700,0.416);
INSERT INTO "rss" VALUES(-68,6.600,0.254);
INSERT INTO "rss" VALUES(-67,6.500,2.997);
INSERT INTO "rss" VALUES(-66,6.400,2.997);
INSERT INTO "rss" VALUES(-65,6.300,2.997);
INSERT INTO "rss" VALUES(-64,6.200,1.322);
INSERT INTO "rss" VALUES(-63,6.100,0.873);
INSERT INTO "rss" VALUES(-62,6.000,0.513);
INSERT INTO "rss" VALUES(-61,5.450,0.873);
INSERT INTO "rss" VALUES(-60,5.000,2.997);
INSERT INTO "rss" VALUES(-59,4.600,0.950);
INSERT INTO "rss" VALUES(-58,4.320,1.208);
INSERT INTO "rss" VALUES(-57,4.013,0.710);
INSERT INTO "rss" VALUES(-56,3.140,0.646);
INSERT INTO "rss" VALUES(-55,2.990,0.611);
INSERT INTO "rss" VALUES(-54,2.790,0.495);
INSERT INTO "rss" VALUES(-53,2.550,0.647);
INSERT INTO "rss" VALUES(-52,2.292,3.775);
INSERT INTO "rss" VALUES(-51,2.000,3.898);
INSERT INTO "rss" VALUES(-50,1.980,3.428);
INSERT INTO "rss" VALUES(-49,1.106,0.124);
INSERT INTO "rss" VALUES(-48,1.000,0.212);
INSERT INTO "rss" VALUES(-47,0.892,0.292);
INSERT INTO "rss" VALUES(-46,0.652,0.364);
INSERT INTO "rss" VALUES(-45,0.587,0.429);
INSERT INTO "rss" VALUES(-44,0.531,0.277);
INSERT INTO "rss" VALUES(-43,0.476,0.222);
INSERT INTO "rss" VALUES(-42,0.429,0.175);
INSERT INTO "rss" VALUES(-41,0.386,0.132);
INSERT INTO "rss" VALUES(-40,0.348,0.094);
INSERT INTO "rss" VALUES(-39,0.313,0.059);
INSERT INTO "rss" VALUES(-38,0.282,0.028);
INSERT INTO "rss" VALUES(-37,0.254,0.025);
```

# Bibliography

Alliance, O. H. (2007). Retrieved November 10, 2010, from Open Handset Alliance FAQ: http://www.openhandsetalliance.com/oha_faq.html

Bahl, P., & Padmanabhan, V. N. (2002). RADAR: An In-building RF-based User Locatin and Tracking System. *INFOCOM 2000. 19th Annual Joint Conference of the IEEE Computer and Communications Societies*, (p. 775).

Chandrasekaran, G., Ergin, M. A., Yang, J., Liu, S., Chen, Y., Gruteser, M., et al. (2009). Empirical Evaluation of the Limits on Localization Using Signal Strength. *6th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks,* (pp. 1-9). Rome.

Cyanogen. (2011, December 7). *HTC Hero (CDMA): Full Update Guide*. Retrieved December 14, 2011, from Cyanogenmod: http://wiki.cyanogenmod.com/wiki/HTC_Hero_(CDMA):_Full_Update_Guide

CyanogenMod. (2011, October). *CyanogenMod*. Retrieved October 31, 2011, from Android Community Rom based on Gingerbread: http://www.cyanogenmod.com

Dutta, K. (2011). *ClockworkMod*. Retrieved October 31, 2011, from ClockworkMod: http://www.clockworkmod.com

Enck, W., Ongtang, M., & McDaniel, P. (2009). Understanding Android Security. *IEEE Security & Privacy*, 50-57.

Fang, S.-H., & Lin, T.-N. (2010, May). Cooperative Multi-Radio Localization in Heterogeneous Wireless Networks. *IEEE Transactions on Wireless Communications*, *9*(5), 1547-1551.

Flosi, S. (2011, March). *March 2011 U.S. Mobile Subscriber Market Share*. Retrieved November 2011, from comScore: http://www.comscore.com/Press_events/Press_Releases/2011/5/comScore_Reports_March_2011_U.S._Mobile_Subscriber_Market_Share

Google. (2010, November). Retrieved January 2011, from Android Developers: http://developer.android.com/index.html

Gould, J., & Hoffman, M. (2010, December). *Army Sees Smart Phones Playing Important Role*. Retrieved from ArmyTimes:

http://www.armytimes.com/news/2010/12/army-smart-phones-for-soldiers-121210w/

Gu, Y., Lo, A., & Niemegeers, I. (2009). A Survey of Indoor Positioning Systems for Wireless Personal Networks. *IEEE Communications Surveys & Tutorials, 11*(1), 13-32.

Gustafsson, F., & Gunnarsson, F. (2005, July). Mobile Positioning Using Wireless Networks. *IEEE Signal Processing Magazine*, pp. 41-53.

Gwon, Y., Jain, R., & Kawahara, T. (2004). Robust Indoor Location Estimation of Stationary and Mobile Users. *INFOCOM*, (pp. 1032-1043).

Hall, D., & Llinas, J. (1997, January). An Introduction to Multisensor Data Fusion. *Proceedings of the IEEE, 85*(1), pp. 6-23.

Hamilton, B. R., Ma, X., Baxley, R. J., & Walkenhorst, B. (2010). Node Localization and Tracking Using Distance and Acceleration Measurements. *Second International Workshop on Cognitive Information Processing*, (pp. 399-404). Tuscany, Italy.

HTC. (2011, June). Retrieved November 7, 2011, from HTC Hero Tech Specs: http://www.htc.com/us/products/hero-sprint?view=1-2&sort=1#tech-specs

Kaemarungsi, K., & Krishnamurthy, P. (2004). Properties of Indoor Recieved Signal Strength for WLAN Location Fingerprinting. *First Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services.*

Kellogg, D. (2011, September 1). *40 Percent of U.S. Mobile Users own Smartphones; 40 Percent are Android*. Retrieved from Nielsen Wire: http://blog.nielsen.com/nielsenwire/online_mobile/40-percent-of-u-s-mobile-users-own-smartphones-40-percent-are-android/

Liu, H., Darabi, H., Banerjee, P., & Liu, J. (2007, November). Survey of Wireless Indoor Positioning Techniques and Systems. *IEEE Transactions on Systems, Man, and Cybernetics, 37*(6), 1067-1080.

Llinas, J., Bowman, C., Rogova, G., Steinberg, A., Waltz, E., & White, F. (2004). Revisiting the JDL Data Fusion Model II. *Seventh International Conference on Information Fusion*, (pp. 1218-1230).

Moore, G. E. (1965, April). Cramming More Components Onto Integrated Circuits. *Electronics Magazine, 38*(8).

Motorola. (2009, October 28). *Motorola Fact Sheet*. Retrieved November 7, 2011, from DROID by Motorola Fact Sheet: http://mediacenter.motorola.com/content/detail.aspx?releaseid=12059&newsareaid=22

Mue, H. (2011, October). *Android-wifi-tether*. Retrieved October 2011, from Wireless Tether for Root Users: http://code.google.com/p/android-wifi-tether/

Pandya, D., Jain, R., & Lupu, E. (2003). Indoor Location Estimation Using Multiple Wireless Technologies. *Personal, Indoor and Mobile Radio Communications*, (pp. 2208-2212). Beijing.

Patwari, N. (2005). *Location Estimation in Sensor Networks.* PhD Thesis, University of Michigan, Ann Arbor.

Patwari, N. (2008). Localization, Cooperative. In H. Xiong, & S. Shekhar (Eds.), *Encyclopedia of GIS* (pp. 616-622). New York: Springer.

Patwari, N., & III, A. O. (2006). *Signal Strength Localization Bounds in Ad Hoc & Sensor Networks When Transmit Powers Are Random.* DARPA Defense Sciences Office under Naval Research, IEEE, University of Michigan, Ann Arbor.

Patwari, N., Ash, J. N., Kyperountas, S., III, A. O., Moses, R. L., & Correal, N. S. (2005, July). Locating the Nodes. *IEEE Signal Processing Magazine*, 54-69.

Patwari, N., III, A. O., Perkins, M., Correal, N. S., & O'Dea, R. J. (2003, August). Relative Location Estimation in Wireless Sensor Networks. *IEEE Transanctions on Signal Processing, 51*(8), 2137-2148.

Rappaport, T. S. (2002). *Wireless Communications: Principles and Pracitce* (2nd ed.). Upper Saddle River, NJ: Prentice Hall PTR.

Sayed, A. H., Tarighat, A., & Khajehnouri, N. (2005, July). Network-Based Wireless Location. *IEEE Signal Processing Magazine*, 24-40.

Shabtai, A., Fledel, Y., Kanonov, U., Elovici, Y., Dolev, S., & Glezer, C. (2010). Google Android: A Comprehensive Security Assessment. *IEEE Security & Privacy*, 35-44.

Steinberg, A. N., & Bowman, C. L. (2009). Revisions to the JDL Data Fusion Model. In M. E. Liggins, D. L. Hall, & J. Llinas, *Handbook of Multisensor Data Fusion: Theory and Practice* (pp. 45-67). Boca Raton, FL: CRC Press.

Sun, G., Chen, J., Guo, W., & Liu, K. R. (2005, July). Signal Processing Techniques in Network-Aided Positioning. *IEEE Signal Processing Magazine*, pp. 12-23.

Tummala, D. (2005). *Indoor Propagation Modeling at 2.4 GHz for IEEE 802.11 Networks.* University of North Texas, Department of Electrical Engineering. University of North Texas.

Xu, Y., Ouyang, Y., Le, Z., Ford, J., & Makedon, F. (2007). Mobile Anchor-free Localization for Wireless Sensor Networks. *International Distributed Computing in Sensor Systems.* Santa Fe.

| REPORT DOCUMENTATION PAGE | *Form Approved* *OMB No. 074-0188* |
|---|---|

| | | |
|---|---|---|
| **1. REPORT DATE** *(DD-MM-YYYY)* 22-03-2012 | **2. REPORT TYPE** Master's Thesis | **3. DATES COVERED** *(From – To)* Aug 2010 – Mar 2012 |

| | |
|---|---|
| **4. TITLE AND SUBTITLE** Cooperative Localization on Computationally Constrained Devices | **5a. CONTRACT NUMBER** |
| | **5b. GRANT NUMBER** |
| | **5c. PROGRAM ELEMENT NUMBER** |
| **6. AUTHOR(S)** Cicale, Randy S, Capt, USAF | **5d. PROJECT NUMBER** N/A |
| | **5e. TASK NUMBER** |
| | **5f. WORK UNIT NUMBER** |
| **7. PERFORMING ORGANIZATION NAMES(S) AND ADDRESS(S)** Air Force Institute of Technology Graduate School of Engineering and Management (AFIT/ENV) 2950 Hobson Way, Building 640 WPAFB OH 45433-8865 | **8. PERFORMING ORGANIZATION REPORT NUMBER** AFIT/GCO/ENG/12-04 |
| **9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)** Intentionally Left Blank | **10. SPONSOR/MONITOR'S ACRONYM(S)** |
| | **11. SPONSOR/MONITOR'S REPORT NUMBER(S)** |

**12. DISTRIBUTION/AVAILABILITY STATEMENT**
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

**13. SUPPLEMENTARY NOTES**
This material is declared a work of the United States Government and is not subject to copyright protection in the United States.

**14. ABSTRACT**
Cooperative localization is a useful way for nodes within a network to share location information in order to better arrive at a position estimate. This is handy in GPS contested environments (indoors and urban settings). Most systems exploring cooperative localization rely on special hardware, or extra devices to store the database or do the computations. Research also deals with specific localization techniques such as using Wi-Fi, ultra-wideband signals, or accelerometers independently opposed to fusing multiple sources together. This research brings cooperative localization to the smartphone platform, to take advantage of the multiple sensors that are available. The system is run on Android powered devices, including the wireless hotspot. In order to determine the merit of each sensor, analysis was completed to determine successes and failures. The accelerometer, compass, and received signal strength capability were examined to determine their usefulness in cooperative localization. Experiments at meter intervals show the system detected changes in location at each interval with an average standard deviation of 0.44m. The closest location estimates occurred at 3m, 4m and 6m with average errors of 0.15m, 0.11m, and 0.07m respectively. This indicates that very precise estimates can be achieved with an Android hotspot and mobile nodes.

**15. SUBJECT TERMS**
Android, Cooperative Localization, RSS, Multi-sensor fusion

| **16. SECURITY CLASSIFICATION OF:** 97 | | | **17. LIMITATION OF ABSTRACT** | **18. NUMBER OF PAGES** | **19a. NAME OF RESPONSIBLE PERSON** Hemmes, Jeffrey M., Maj, Ph.D, USAF |
|---|---|---|---|---|---|
| **a. REPORT** | **b. ABSTRACT** | **c. THIS PAGE** | | | **19b. TELEPHONE NUMBER** *(Include area code)* |
| U | U | U | UU | 98 | (937) 255-6565, x 4619 (Jeffrey.hemmes@afit.edu) |